电子信息与电气工程技术丛书 E&E

HETEROGENEOUS NETWORK CONVERGENCE AND PROGRAMMABLE TECHNOLOGY



异构网络融合与可编程技术

魏翼飞 宋梅 著

Wei Yifei Song Mei

清华大学出版社

电子信息与电气工程技术丛书

异构网络融合与可编程技术

魏翼飞　宋　梅　著

清华大学出版社
北　京

内 容 简 介

本书系统而又深入地论述了异构网络融合与可编程的相关知识和关键技术。全书共分 10 章，内容包括异构网络融合与可编程网络相关技术的发展，异构无线网络融合的方案，异构无线网络融合的关键技术，网络可编程的基本概念与关键技术，软件定义网络基础概念与整体架构，基于 SDN 的异构网络融合技术，网络虚拟化的发展背景、发展动因及主要应用，网络功能虚拟化的架构、关键问题和应用场景，异构无线网络仿真平台的相关技术与实现，SDN 仿真平台的搭建，搭建异构网络管理平台。

本书可供从事下一代通信网络研究的专业技术人员、管理人员，特别是网络融合和可编程及虚拟化领域的专业研究人员和工程技术人员阅读，也可作为高等院校从事相关课题研究的师生的参考书。

图书在版编目(CIP)数据

异构网络融合与可编程技术/魏翼飞，宋梅著. —北京：清华大学出版社，2017
(电子信息与电气工程技术丛书)
ISBN 978-7-302-46535-5

Ⅰ. ①异…　Ⅱ. ①魏…　②宋…　Ⅲ. ①异构网络－研究　Ⅳ. ①TP393.02

中国版本图书馆 CIP 数据核字(2017)第 030369 号

责任编辑：盛东亮
封面设计：李召霞
责任校对：李建庄
责任印制：刘海龙

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 装 者：三河市铭诚印务有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：26　　**字　　数**：633 千字
版　　次：2017 年 7 月第 1 版　　**印　　次**：2017 年 7 月第 1 次印刷
印　　数：1～2000
定　　价：79.00 元

产品编号：073294-01

前言

在技术进步、市场竞争和用户需求的共同推动下，信息通信技术特别是无线通信技术的发展突飞猛进，无处不在的网络服务将成为现实。在蜂窝移动通信技术向宽带化方向快速演进、力争提供无线宽带服务的同时，各种无线接入技术不断成熟并普及，促使宽带业务向无线化发展，开始提供部分移动通信功能，试图通过宽带移动化进入移动通信市场。这两大分支在相互竞争独立发展的同时相互借鉴、取长补短，满足了终端和业务的多样化、个性化发展需求。不同网络架构、不同组网技术、不同协议软件、不同服务质量、不同管理模式的多种网络共存，共同为用户提供泛在网络服务的局面将长期存在。

各种信息通信网络在高速发展的同时，也日益显现出受网络封闭与僵化特性的制约，导致异构网络之间难以融合互通、难以优化网络整体性能、难以提供定制化的服务，并且多种网络形态共存已经导致了过度服务、资源和能耗消耗过大的问题。如何将这些异构网络融合在一起，互联互通，并且多网协同工作，提供覆盖广、带宽高、移动性强且费用低廉的接入服务，节约资源、适度服务将是未来通信系统的发展方向。在异构网络场景下引入可编程思想，对传统网络架构进行改进，通过承载与控制分离或者网元功能软件化，提升网络的软件化程度和可编程能力，实现网络的灵活可控制、融合可演进、开放可编程以及弹性可定制的特性，为未来信息通信网络发展提供了更大的发展空间。

基于软件定义与可编程技术的异构网络融合方案能够提高融合网络整体性能、最大化网络资源利用率、降低网络能耗、支持无缝切换和漫游、提供智能最佳接入及服务质量和安全保障，适应终端和业务的多样化、个性化发展。对运营商而言，传统网络昂贵的专业设备被通用的可编程交换设备和控制器所取代，能够有效降低其组网成本、提供按需分配的适度服务；对于互联网企业而言，软件定义网络架构屏蔽了低层交换设备的差异性，网络管理者不需要通过各种协议来实现复杂的网络功能，仅仅调用软件接口就可以完成对网络的可重构管理和控制，提高了网络管理效益；对普通用户而言，由于可编程网络能够根据业务需求对网络进行实时动态调整，因此可以提供给用户更好的服务体验。通过集中管理和协同控制，实现网络资源的高效利用；通过网络虚拟化，实现客户定制化网络；通过开放的控制与数据平面接口，实现新技术的快速应用。异构网络融合与可编程技术将打破传统网络路径选择与资源优化瓶颈，增强网络连通性，降低异构网络融合复杂度，推动信息通信网络向智能可编程、资源最优化、绿色高能效的方向发展演进。

本书着眼于全面、系统、深入地介绍异构网络融合与可编程的相关知识和关键技术，分析总结了当前异构网络融合及可编程相关技术的发展和研究现状，结合课题组近年来的科研成果，介绍了基于软件定义的异构网络融合技术以及异构网络融合和可编程仿真平台及实验平台的搭建。

全书分为10章。第1章是全书的概要，首先介绍无线通信技术与有线通信技术的发展历程，然后介绍了异构网络融合与可编程网络相关技术，最后探讨了未来网络的特

前言

点。第 2 章介绍异构无线网络融合的方案，分别对异构无线网络业务层融合、IP 层融合、物理层融合、组网技术的融合等方案进行了详细阐述。第 3 章分析了异构无线网络融合的关键技术，对异构网络融合过程中需要考虑的 4 类关键问题进行深入分析。第 4 章阐述了网络可编程的基本概念、关键技术与发展现状，并重点对可编程网络中软件定义特性在无线网络与数据中心中的应用进行详细的介绍和分析。第 5 章首先介绍软件定义网络基础概念与整体架构，随之分别对其数据平面与控制平面以及相关接口进行了详细阐述。第 6 章概述基于 SDN 的异构网络融合技术，介绍了 SDN 在不同网络场景中的应用，以及如何进行异构网络间的选择与切换，讨论基于 SDN 进行网络资源管理的方法，并针对网络能耗问题提出基于 SDN 的节能方案，给出了当前 SDN 在网络中的部署方案。第 7 章分为两部分，首先介绍网络虚拟化的发展背景、发展动因、主要应用，其次对网络功能虚拟化的架构、关键问题和应用场景等进行了详细阐述。第 8 章总结了异构无线网络仿真平台的相关技术与实现，简要介绍 NS-2 网络模拟器及仿真分析所需要的相关工具，对支持 UMTS/WLAN 双模的网络仿真平台及相关仿真实例的实现进行详细阐述。第 9 章介绍 SDN 仿真平台的搭建，通过 Mininet 工具来定义网络，使用了开源的 Opendaylight，在 Mininet 和 Opendaylight 之间搭建了 FlowVisor，通过对网络流量关键字段的划分，使得同一物理网络中可以部署多个控制器来完成不同的策略。第 10 章围绕如何搭建异构网络管理平台展开，首先介绍现有网络配置管理技术的应用与实现原理，接下来就如何搭建异构网络管理平台以及搭建平台所涉及的关键技术进行详细阐述。

本书的编写考虑到了不同层次读者的需要，全面地介绍了异构网络融合及可编程相关技术的概念、设计思路和关键技术，包括各种通信技术的发展趋势、网络融合方案及融合的关键技术、网络可编程技术、软件定义网络、基于 SDN 的异构网络融合技术、网络虚拟化与网络功能虚拟化、异构网络融合和可编程仿真平台及实验平台的搭建，便于读者对未来可编程异构融合网络形成系统全面的知识体系。读者也可以根据自身需要，有选择性地阅读相关的章节。

本书的编撰工作凝结了北京邮电大学 ICN&CAD 中心张勇、满毅、王莉、滕颖蕾、刘洋、郭达、王小娟等教师、博士生及硕士生的研究成果，李俏、李莉、屈银翔、顾博、侯永福、王昊、赵梦雨、吴凯丽、贾子寒、彭翀霄等参与了本书中重要部分的编写工作，在这里特别表示感谢。加拿大卡尔顿大学 Fei Richard YU 教授、爱尔兰都柏林城市大学 Xiaojun WANG 教授、美国休斯敦大学 Zhu HAN 教授对本书提出了很好的建议，在此向他们表示衷心的感谢。此外，还要感谢国家自然科学基金项目(61571059)对相关研究的资助。

由于编者水平和视野所限，以及编写时间仓促，加之信息通信技术发展日新月异，书中难免有疏漏甚至错误之处，恳请读者批评指正。

编著者

2017 年 4 月

目录

目录

目录

目录

目 录

第1章 信息通信技术发展趋势

本章是全书的概要，首先介绍无线通信技术，包括移动通信与无线接入技术的起源与发展历程；然后介绍有线网络，包括互联网、有线电视网络以及光网络；在简单介绍了无线通信与有线网络技术发展概况后，讨论异构网络融合与可编程网络；最后，根据信息通信技术的发展趋势，探讨未来网络的特点。

1.1 无线通信技术发展

无线通信是通信行业中发展最快、最活跃的一个领域，已经获得媒体与公众的广泛关注。蜂窝移动通信系统的业务量在最近十几年呈指数式增长，世界范围的移动接入数已超过72亿。实际上，在大多数发达国家，蜂窝电话（手机）已经成为工作以及生活的一部分，随着蜂窝移动通信技术的更新换代，如今的智能手机已远远超出"电话机"的范畴，它相当于一台小型计算机，能够实现许多难以想象的功能；在一些发展中国家，移动电话正在快速取代旧式的有线系统。此外，无线局域网络（wireless local area networks，WLAN）作为有线网络的补充，部署在一些居民区、商业区以及校区，以其灵活方便的接入方式和低成本的网络部署，在一些地区甚至取代有线网络。一些新的无线通信系统应用，包括无线传感器网络、智能家居、高速路与工厂自动化、远程医疗等正在由概念研究向实际系统应用转变。无线系统的迅猛发展与智能终端设备的快速普及相互促进，预示着无线网络发展的广阔前景。然而，在建设能够承载众多新兴应用的健壮的无线网络的过程中，仍然存在着技术挑战。下面介绍移动通信与无线接入技术的发展历程。

1.1.1 移动通信系统的发展

移动通信以其"移动"的本质区别于传统静态的固定式通信，其产生具有革命性意义。针对传统固定通信的全封闭式传输线路的限制，无线通信以开放式传播来传递信息，将通信方式从静态推广至可移动

式的准动态。移动通信则是在无线通信的基础上，进一步引入了用户的移动性，从而使终端从可移动的准动态进一步发展到真正的全动态[1,2]。1928年，美国普度大学学生发明了工作于2MHz的超外差式无线电接收机，并很快在底特律的警察局投入使用。这是世界上第一种可以有效工作的移动通信系统，但是这种移动通信系统的工作频率较低、话音质量差、自动化程度低，难以与公众电话网络互通。到20世纪50年代，美国和欧洲部分国家相继成功研制了公用移动电话系统，在技术上实现了移动电话系统与公众电话网络的互通，并得到了广泛的使用。然而这种公用移动电话系统仍然采用人工接入方式，系统容量较小。从20世纪60年代中期至70年代中期，美国推出了改进型移动电话系统，它使用150MHz和450MHz频段，采用大区制、中小容量，实现了无线频道自动选择及自动接入公共电话网。20世纪70年代中后期，随着民用移动通信用户数量的增加、业务范围的扩大，有限的无线频谱与不断增长的容量要求之间的矛盾日益尖锐。为了更有效地利用无线频谱资源，美国贝尔实验室提出了在移动通信发展史上具有里程碑意义的小区制、蜂窝组网理论，通过在非相邻小区进行频谱再用（frequence reuse，又称频谱复用）的方式大幅扩充了网络容量，为移动通信系统在全球广泛应用开辟了道路。

1. 第一代移动通信系统

第一代移动通信系统（the first-generation wireless telephone system，1G）采用了蜂窝组网、模拟调制和频分多址接入（frequency division multiple access，FDMA）等技术，支持话音和低速率的数据通信，是第一个真正意义上的移动通信网络，于20世纪80年代初开始商用。严格来说，1G是模拟或半模拟移动网络，模拟无线信道但是采用数字交换。这一代移动通信系统有多种制式，如AMPS、NMT、TACS、C-450、Radiocom2000、RTMI等。利用蜂窝结构对空间的有效利用性，模仿蜂窝的正六边形结构特征将覆盖范围划分成多个更小的单元，与大区制通信原理相同，只需保证每个小区内及相邻小区的频率不重复使用，所以在更短的距离内可以实现频率的复用，从而大大提高了频谱利用率。通过适当的小区与频率复用规划显著提高网络容量的同时，也带来越区切换的问题。一个小区覆盖范围越小，用户移动越快，信道的切换次数就越多，交换中心的控制交换技术也就越复杂。

第一代移动通信系统经历的时间大约由1980年至1994年，主要采用模拟技术和频分多址接入，实际数据速率为2.4Kb/s。这一代移动通信系统有很多不足之处，例如多种制式并存且互不兼容，容量有限，通话质量不高，不能提供数据业务等。此外，由于传输带宽的限制，第一代移动通信系统不能进行长途漫游，因此，只是一种区域性移动通信系统。

2. 第二代移动通信系统

20世纪90年代，数字技术的成功应用以及其优于模拟技术的性能，极大地推动了移动通信技术的发展，第二代移动通信系统是第一个数字式蜂窝移动通信系统。2G标准主要有欧洲的全球移动通信系统（global system for mobile，GSM）、美国的IS-136（也叫做D-AMPS）和IS-95（也叫做cdmaOne）、日本的个人数字蜂窝系统（personal digital

cellular，PDC）。其中世界范围内应用最为广泛的 GSM，是为了解决欧洲第一代移动通信系统制式多、不兼容的问题而发展起来的，于 1991 年投入欧洲市场。

相对于 1G，2G 最鲜明的特色是数字化，采用时分多址（time division multiple access，TDMA）（GSM）、码分多址（code division multiple access，CDMA）（IS-95）方式实现对用户的动态寻址功能，并以数字式蜂窝网络结构和频率（相位）规划实现载频（相位）再用，从而扩大覆盖范围并满足用户数量增长的需求。这些蜂窝系统及其标准最初是为电话业务开发的，核心网以电路交换的方式提供服务，虽然能够提供低速率数据服务，但是其数据速率和延时实质上是由电话业务需求所决定。通用分组无线服务（general packet radio service，GPRS）是从 GSM 系统基础上发展起来的分组无线数据业务，又称为 2.5G，是欧洲电信标准化协会（european telecommunications standards institute，ETSI）从 1993 年开始制定并于 1998 年完成的。GPRS 与 GSM 公用频段、公用基站并共享 GSM 系统与网络中的设施，在保持 GSM 的电路交换方式处理话音业务的同时，增加了分组方式处理数据，拓展了 GSM 业务的服务种类，在核心网部分逐渐实现了从电路交换到电路交换与分组交换并存的转变。为了进一步提高数据业务传输速率，在 GSM/GPRS 的基础上，ETSI 进一步提出了增强型数据速率 GSM 演进技术（enhanced data rate for GSM evolution，EDGE），又称为 2.75G。EDGE 从 GPRS 平滑演进，最大限度地利用了 GSM/GPRS 已有技术以及设备，分别在分组交换业务和电路交换业务方面对 GSM/GPRS 进行了必要的扩展，形成了 EGPRS（enhanced GPRS）和 ECSD（enhanced circuit switch data），在充分利用 GSM 已有资源的情况下，具有三倍于 GPRS 处理数据的能力。

3. 第三代移动通信系统

由于第二代移动通信系统的巨大成功，用户需求迅速增长，网络容量的有限性逐渐明显，无法满足用户需求。第三代移动通信系统（3G）正是顺应需求而产生，在二代网络基础上进行演进与升级。国际电信联盟（international telecommunication union，ITU）1985 年提出 IMT-2000（international mobile telecom system-2000）作为对 3G 性能的要求，对 3G 技术发展与标准化起到了积极的推动作用，3G 标准的技术细节主要由第三代合作伙伴计划 3GPP 和 3GPP2 两大标准组织完成。主要的 3G 标准有 UMTS、WCDMA、CDMA2000 以及 TD-SCDMA。UMTS、WCDMA 从 GSM/GPRS 网络过渡升级而来，核心网部分是平滑过渡，空中接口部分则是革命性变化。以 IS-95 为代表的 CDMA ONE、CDMA2000 1X 等均是美国高通公司研制和开发的国际性移动通信标准，并逐渐演进到 CDMA2000，它们在制式与网络结构方面是一脉相承的。TD-SCDMA 是由我国提出来的，在最初 SCDMA 技术的基础上发展而来，继承了 GSM 核心网技术，是基于智能天线的同步 CDMA 系统，同时还采用了联合检测和软件无线电等多项技术。

除了传统的语音服务，3G 旨在融合蜂窝移动通信和因特网（Internet），从而为用户提供不同种类的数据业务，在终端移动与传输线路随终端移动的基础上进一步引入了数据业务类型动态选择特性。3G 核心网包括电路交换域、分组交换域和 IP 多媒体系统（IP multimedia subsystem，IMS）三个主要部分。其中，IMS 被认为是 3G 网络的核心技

术，也是解决移动与固网融合，引入语音、数据、视频三重融合等差异化业务的重要方式。

4. 第四代移动通信系统

第四代移动通信系统（4G）将3G中的业务类型动态选择特性全面增强，同时引入网络拓扑与网络运行的动态性，将3G与WLAN融合为一体，为移动用户提供高速数据业务，如高清图像业务、会议电视、虚拟业务等。2012年，国际电信联盟无线通信全会通过了高级国际移动通信（IMT-Advanced）标准，即4G标准。三个4G标准分别基于3GPP的长期演进（long term evolution，LTE）技术和全球微波互联接入（worldwide interoperability for microwave access，WiMAX）技术。欧洲提交的FDD-LTE-Advanced和我国提交的TD-LTE-Advanced，都是基于LTE技术演进而来且完全兼容LTE，WirelessMAN-Advanced（IEEE 802.16m）则是WiMAX的后续研究标准。

4G关键技术包括正交频分复用（orthogonal frequency division multiplexing，OFDM）、增强型多入多出天线（multiple input multiple output，MIMO）、载波聚合、智能天线、软件无线电、中继技术、家庭基站、基于IP的核心网等。4G网络结构包括演进型的接入网和核心网，如图1-1所示。接入网部分的基站为演进型NodeB，即eNodeB，相比3G中的NodeB，集成了部分RNC的功能，减少了通信时协议的层次。同时在接入网部分增加了中继站Relay Node，将一条基站—用户设备链路分割为基站—中继站和中继站—用户设备两条链路，从而能够将一条质量较差的链路替换为两条质量较好的链路，以扩大网络覆盖以及小区边缘吞吐量。核心网部分，MME（mobility management entity）为控制平面服务，处理用户接入中与移动性和安全性有关的信息。HSS（home subscriber server）是存储与用户以及订阅有关的信息的数据库，起到移动性管理的功能。Serving Gateway和PDN Gateway为用户平面服务，传输用户设备与外部网络之间的IP数据。PCRF（policy and charging rules function）维护网关（gateway）控制会话及IP连接访问网络会话之间的关联，负责IP业务数据流的策略与计费控制。

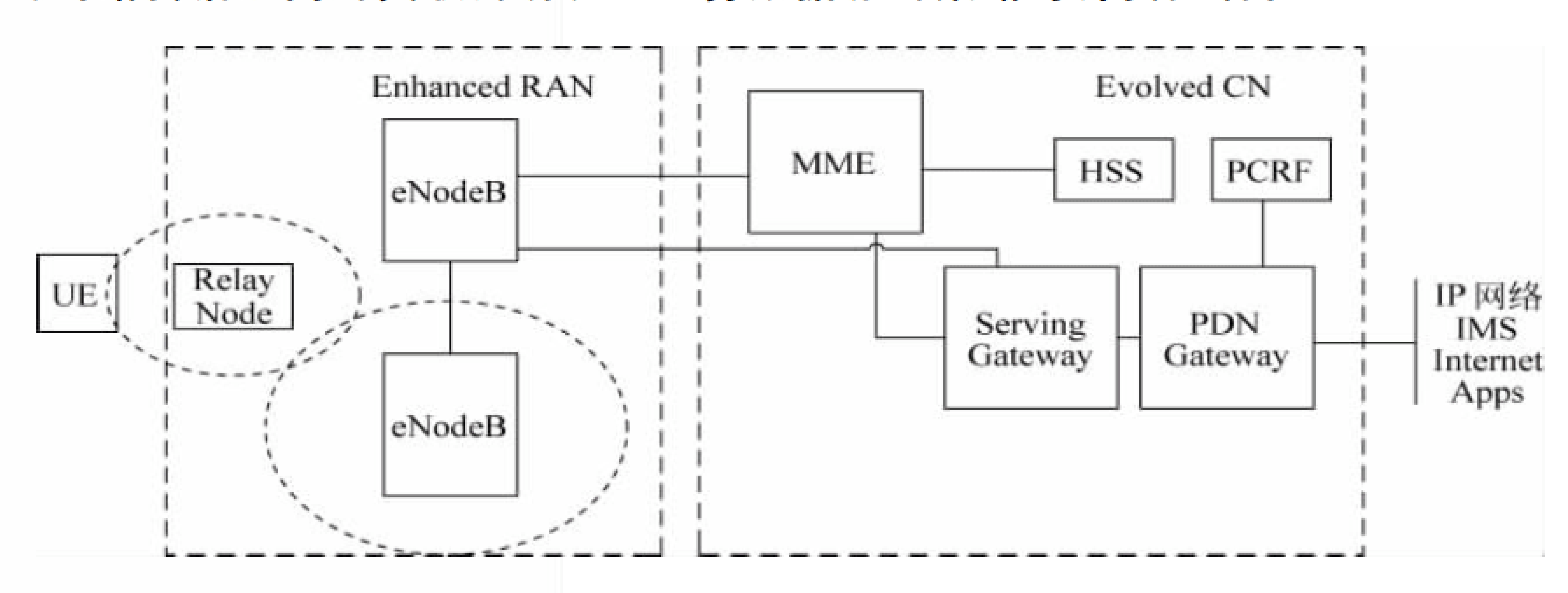


图1-1　4G网络结构

从2G到4G的演进过程中，核心网部分由2G中只由电路交换承载话音业务以及少量数据业务，到目前4G中由分组交换承载包括话音、数据在内的所有业务，逐渐实现了基于IP的移动通信网络结构，如图1-2所示。移动通信网络逐渐演进为从接入网到核心网全IP化网络。



图 1-2 移动通信系统核心网演进过程

5. 第五代移动通信系统展望

未来移动通信的发展一方面取决于用户需求，另一方面则受到实现时所面临的环境与条件的限制。思科年度 VNI(visual network index)报告中的数据显示[3]，由于智能手机、平板电脑等终端设备的普及，无线数据业务量在全球范围内仍在持续快速地增长，仅仅通过对现有移动通信网络进行改进，将无法满足未来所面临的需求。未来第五代移动通信系统(5G)将如前四代移动通信系统中的每一代一样具有根本性改变和性能的大幅提升，具体体现在：超高载波频率超宽带宽、超密集的基站与用户设备以及前所未有的天线数量等。与前四代移动通信不同之处在于，5G 将是高度一体化的技术集合与前所未有的用户体验：将新的 5G 空中接口、频谱和 4G 与无线局域网结合在一起，提供普遍的高数据率的覆盖与无缝切换的用户体验。为了实现这些，核心网络也将达到空前的灵活性与智能化，频谱管理策略需要被重新思考与改进，能量与成本消耗方面的考虑将会变得更加重要。

数据率的大幅提升将很有可能通过以下方式实现：增加带宽，主要通过向毫米波频段转移以及更有效地使用 WiFi 5GHz 的非授权频段，一起实现更多的频带；提升频谱效率，主要通过 MIMO 技术的进步实现，提升每个网络节点单位频段的数据率(即 b/s/Hz)。除了数据的提升，5G 网络还必须降低延迟，降低能量消耗与成本，支持一些低速率连接，为了满足这些要求，一些重要研究正在进行：物理层方面，一些尝试能够替代 OFDM 的多址接入技术已被提出[4]，如 GFDM(generalized frequency division multiplexing)；基于云与虚拟化的网络结构的研究；提高能量效率的资源分配方式、网络设计方法、可再生能源供电以及设备节能策略；频谱分配政策、行业标准以及经济上的考虑等非技术性问题。总之，在通往 5G 的路上，将会存在跨越协议栈各个层面的技术挑战以及政策、商业上的非技术问题，这也为研究者们提供了想象与创新的空间。

在 40 多年的时间里，移动通信系统经历了 4 个代际演进，如图 1-3 所示。1G 满足了用户基本的通话与移动需求，而模拟系统容量小、各标准互不兼容等明显的不足使其很快被以数字系统为主要特点的 2G 取代。2G 满足用户漫游的高级移动需求，提供数据业务，其标准化面向全球。3G 为用户提供无缝漫游、全球无线接入、高数据率的服务。4G 融合了移动通信与无线宽带接入技术，提供基于 IP 的非常高速率的语音、数据业务服务。然而，随着物联网高速发展的潜力不断释放，移动通信系统需要用有限的网络资源与容量去容纳指数级增长的负载量，未来 5G 移动通信系统将会以高度的通用性、可扩展性以及高效率来满足用户不断变化的多种需求，最终达到超高数据速率、低延迟、广泛接

入、低能耗、最佳用户体验以及高可靠性、安全性等目标。面对不断变化的网络环境所带来的挑战与机遇，包含用户、运营商、设备制造商等在内的移动生态系统在未来将产生深入变革并取得进一步发展。

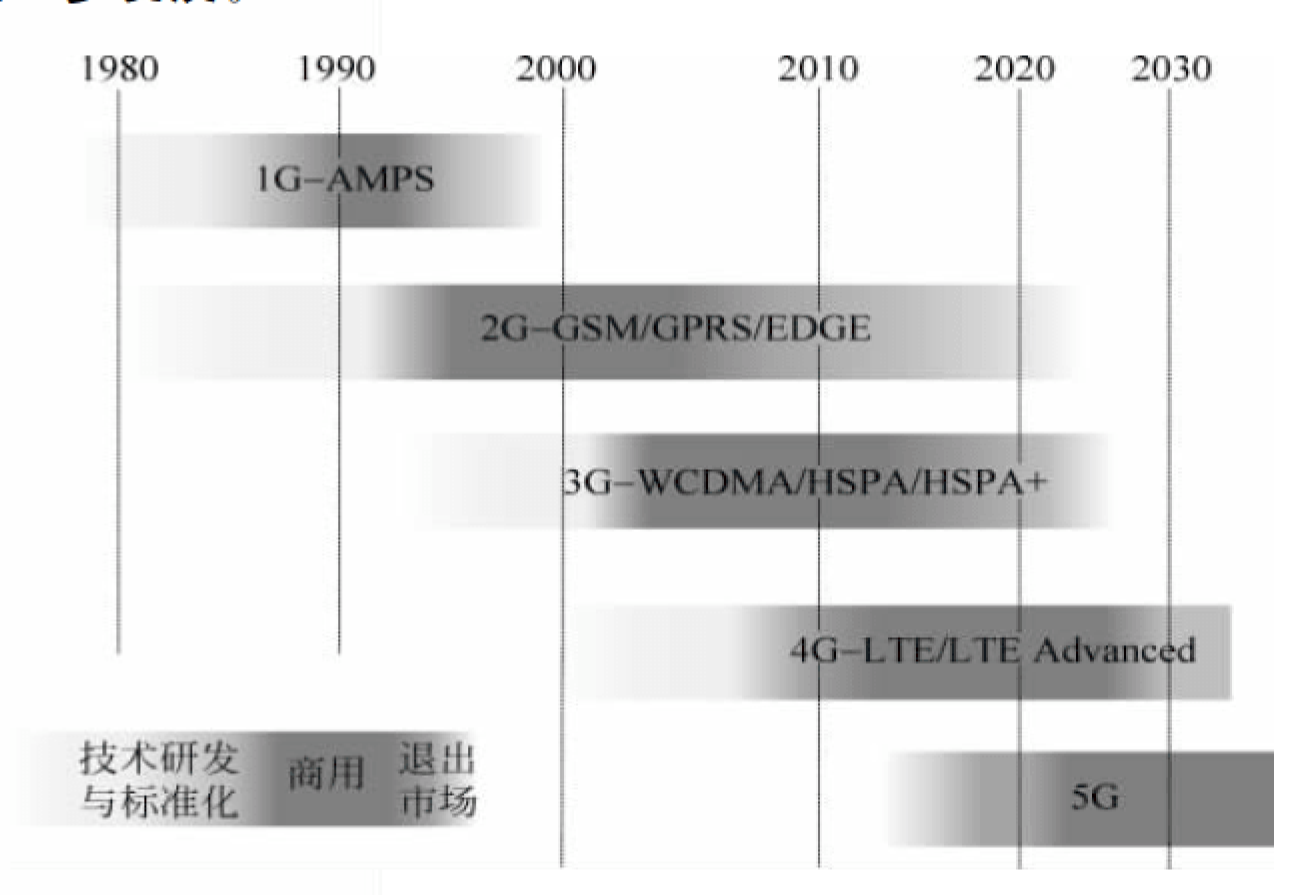


图 1-3　五代移动通信系统演进过程

1.1.2　无线接入技术的发展

随着移动通信技术的迅速发展，仅仅发展了几十年的移动电话数已经超过了发展历史一百多年的固定电话数。人们对于移动通信的需求也反映在计算机网络中，希望能够在移动中使用计算机网络。1971 年 6 月，由美国夏威夷大学所开发的实验性电脑网络系统 ALOHA 网络(ALOHAnet)开始成功运作，这是世界上第一个无线通信计算机网络。本节主要介绍无线个域网、无线局域网(其中包含有固定基础设施与无固定基础设施的无线局域网两大类)以及无线城域网。

1. 无线个域网

无线个域网(wireless personal area network，WPAN)是在个人工作生活的地方把个人使用的电子设备(例如便携式电脑、智能手机以及打印机等)通过无线技术连接起来的网络，所覆盖的范围大约在 10m 左右。WPAN 实际上是一个小功率、小范围、低速率和低价格的电缆替代技术。IEEE、ETSI、ITU 和 HomeRF 等组织都曾致力于 WPAN 技术标准的开发与制定。其中，IEEE 802.15 工作组由 IEEE 于 1998 年成立，是 WPAN 规范标准研发与制定的主要组织，制定的标准包括 802.15.1～802.15.5 等，主要定义了物理层与 MAC 层中的服务和协议。目前比较典型的 WPAN 技术主要有蓝牙、超宽带、ZigBee、IrDA 与近场通信等，它们具有各自的特点并且满足不同种类的服务。下面将分别介绍这几种主要的 WPAN 技术。

(1) 蓝牙一直以来作为 WPAN 应用的主流技术被大众广为熟知。蓝牙标准是在 1998 年由诺基亚、IBM、爱立信等公司共同推出的，即后来的 IEEE 802.15.1 标准。蓝牙技术为移动设备或固定设备之间的通信环境建立通用的无线空中接口，将通信技术与计

算机技术结合起来，使通信产品、计算机产品和消费类电子产品在没有电线或电缆相互连接的情况下，能够在近距离范围内实现相互通信或操作。蓝牙技术具有以下特点：随时随地用无线接口来替代有线电缆进行连接，可以应用于多种通信场合的强移植性，较低的功耗、较低的成本并且易于推广。

(2) 超宽带(ultra wideband，UWB)即 802.15.3a 技术，是一种超高速的短距离无线无载波通信技术，也称为脉冲无线电(impulse radio)、时域(time domain)或无载波(carrier free)通信，利用纳秒至微微秒级的非正弦波窄脉冲传输数据，专为在便携式多媒体装置之间传送数据而制定。例如，使用数码摄像机拍摄的高清图片以及视频节目可以不用连接线就复制到便携式电脑中，会议厅中便携式电脑可以不用连线就能通过投影仪投放到大屏幕上。

(3) ZigBee 是一种短距离、低功率、低速率的无线接入技术。ZigBee 标准作为 IEEE 802.15.4 的扩展集，由 ZigBee 联盟定义应用层与网络层，IEEE 802.15.4 工作组定义 MAC 层和物理层，是关于组网、安全和应用软件等方面的技术标准。ZigBee 主要定位于已有的标准还不能满足需求的特定市场，适合用于自动控制和远程控制领域，可以嵌入各种设备，应用领域包括家庭和楼宇网络、工业控制、商业、公共场所、农业控制、医疗等。

(4) IrDA 是红外数据组织(infrared data association)的简称，目前广泛采用的 IrDA 红外连接技术就是由该组织提出的。IrDA 协议栈是一套专门针对点对点红外通信的协议，其核心协议包括红外物理层，定义了硬件要求和低级数据帧结构以及帧传送速度；红外链路建立协议，在自动协商好的参数基础上提供可靠的、无故障的数据交换；红外链路管理协议，提供建立在红外链路连接上的多路复用及数据链路管理；信息获取服务，提供一个设备所拥有的相关服务检索表。

(5) 近场通信(near field communication，NFC)是一种新兴的短距离高频无线通信技术，它是一种由射频识别(radio frequency identification，RFID)技术演变而来并与之兼容的非接触式识别和互联技术。NFC 技术将非接触读卡器、非接触卡和点对点功能整合到一块 NFC 芯片上，使得 NFC 设备可以进行双向的通信，支持主动通信和被动通信两种通信模式。NFC 广泛兼容非接触卡领域的相关协议，并且形成了自己的技术标准，NFC 终端需要支持多种协议。NFC 技术是由当时的飞利浦半导体(现为恩智浦半导体公司)、诺基亚和索尼共同研发并于 2004 年前后推出的，而后业界创建了一个非营利性的标准组织 NFC Forum 以促进 NFC 技术的实施和标准化工作。然而，因为缺乏合适的应用模式，NFC 在推出之后的很长一段时间之内并未占据市场。近年来随着智能手机的普及以及移动支付的发展，NFC 受到了越来越多的关注，各种应用和产品层出不穷，例如基于 NFC 的移动钱包等。

WPAN 技术得到了飞速的发展，蓝牙、UWB、ZigBee、NFC 等各种技术接连被提出，它们都有各自的特点，或基于传输速度、距离、功耗的特殊要求，或着眼于功能的扩展性，或满足某些单一应用的特别要求等。但是没有一种技术可以满足所有的要求，为了满足用户需求的多样性，就需要多种 WPAN 技术共同为用户提供服务。

2. 无线局域网

无线局域网(wireless local area networks，WLAN)在局域范围内通过电磁波传送和接收数据，利用无线方式进行通信，代替了常规固定局域网中使用的双绞线、同轴线路或

光纤的通信方式。由于摆脱了通信线路的束缚，无线局域网技术具有传统局域网无法比拟的灵活性，相对于有线网络，无线局域网的组建、配置和维护较为容易。无线局域网分为有固定设施的和无固定设施的两大类。有固定设施的无线局域网包含接入点（基站）和空中接口，空中接口也被称为用户接口，基站采用的空中接口与用户设备的接口兼容。此外，由多个基站构成的网络通常包含交换机和控制器，它们的功能是将数据业务传递给某一个特定的基站，各个基站可以通过网络远程交换接入点配置信息，或者检测无线网络上的业务情况。无线局域网发展初期技术标准并不统一，主要有美国的 HomeRF、ETSI 提出的 HiperLAN 标准以及 IEEE 802.11 系列标准。然而在技术与市场的激烈竞争中曾经的 HomeRF 逐渐退出了无线局域网主流市场，目前市场上的无线局域网标准主要是 IEEE 802.11 系列标准，ETSI 和 IEEE 的标准可以互操作。

IEEE 802 工作组制定的标准主要涉及物理层和 MAC 这两个层面。1991 年，802.11 工作组正式成立，同年发布了 802.11 标准。该标准在底层的物理层提供了两种解决方案：调频扩频技术（FHSS）和直接序列扩频（DSSS）。它们是基于开放的 2.4GHz 频段的，并且这两种机制都提供了基本的 1Mb/s 传输速度和可选的 2Mb/s 模式。MAC 层机制基于"监听"，即著名的分布式协调功能（distributed coordination function，DCF），应用了载波监听多路访问/冲突避免（CSMA/CA）策略，不同于 802.3 中的冲突检测，该机制能够更加主动地避免碰撞的发生。802.11 使用星形拓扑，如图 1-4 所示，其中心叫做接入点（access point，AP），可以用于办公室局域网和校园网中用户终端的无线接入。

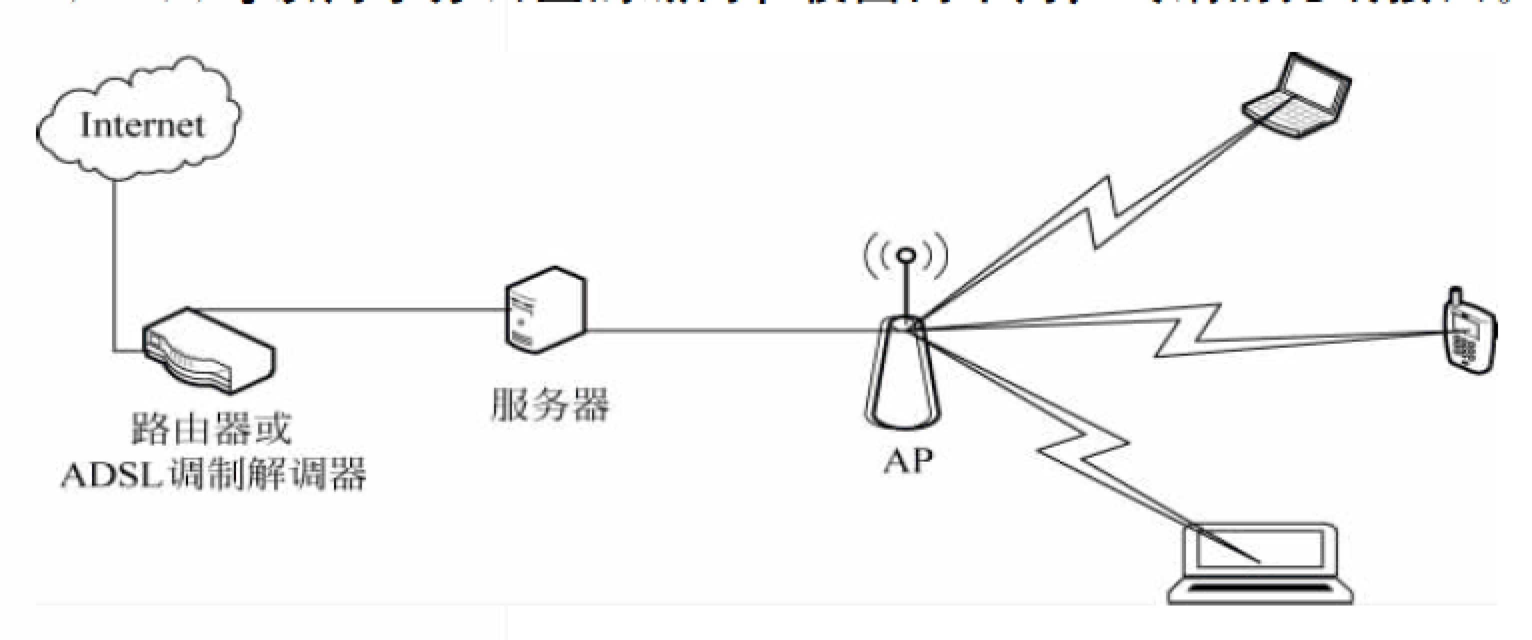


图 1-4　WLAN 星形结构

使用 802.11 系列协议的 WLAN 又称为 Wi-Fi（wireless-fidelity，无线保真度），因此在一些文献中，Wi-Fi 几乎成为 WLAN 的同义词。随着无线局域网市场潜力的进一步发掘，IEEE 802.11 工作组也在最初版本 802.11 标准的基础上制定了一系列修正案以适应市场的要求。802.11 系列标准各有特点，在所用频段以及物理层和 MAC 层技术上各有差异，有各自的应用场景，标准的具体信息可以在 802.11 工作组官方网站 http://ieee802.org/11/查询。IEEE 802.11 系列标准是目前无线局域网最常用的传输协议，各大通信公司都生产基于该标准的无线网卡产品。

3. 无线城域网

无线城域网（wireless metropolitan area network，WMAN）覆盖的区域介于局域网和广域网之间，是一种跨越城市范围的无线网络。移动通信技术的发展已经让人们体会到在大范围移动中无线接入网络给生活和工作带来的便利，同时也希望得到更高的数据

率，无线城域网正是为了满足用户这一需求而得以发展起来。WMAN 可以提供“最后一公里”宽带无线接入，很多情况下可以替代现有的有线宽带接入。1999 年由 IEEE 成立的 802.16 委员会，工作内容就是开发制定无线城域网标准，2001 年 12 月通过了 IEEE 802.16 无线城域网空中接口标准。2003 年由 802.16 技术的部分领先供应商发起的全球微波接入互操作性联盟(worldwide interoperability for microwave access，WiMAX)成立，主要宗旨是推广 IEEE 802.16 的 WMAN 标准。

IEEE 802.16 标准针对传输频率在 10～66GHz 的单载波调制模式，由于工作波长较短，其覆盖在视距范围内。协议规定了空中接口的物理层和 MAC 层两层结构。其中，物理层定义了频率带宽、调制模式、纠错编码、时分复用以及发射机与接收机同步等方面的标准，并采用了按需分配多址(DAMA)和时分多址(TDMA)接入技术。MAC 层又分为业务特定汇聚、公共部分和加密三个子层。业务特定汇聚子层接收来自高层的业务数据，并将这些数据封装成特定的业务包；公共部分子层规定了系统接入、成帧、带宽分配以及连接的建立与维护等过程；加密子层则规定了认证、安全密钥交换以及封装协议。

到目前为止，IEEE 802.16 标准系列包括 802.16、802.16a、802.16c、802.16d、802.16e、802.16f、802.16g、802.16h、802.16i、802.16j 等标准，具体信息可参见 802.16 工作组官方网站 http://ieee802.org/16/。各个标准针对特定的应用而制定，各有特点并适用于不同的场合。

1.1.3 无基础设施的无线网络

前面介绍的移动通信网络与无线接入网络都是有固定通信设施(如基站)的无线网络，本节介绍在不方便或者不需要部署固定通信设施的情况下的无线网络技术，主要有移动自组网络、无线传感器网络以及无线 Mesh 网络。与有固定通信设施的无线网络相比，无基础设施的无线网络有其自身的特点以及在网络通信中的特有问题。

1. 移动自组网络

移动自组网络(Ad hoc 网络)是由一些处于平等状态的移动节点之间相互通信组成的临时网络，节点兼备主机和路由器两种角色。一方面，节点作为主机运行相关的协同应用程序；另一方面，节点作为路由器需要运行相关的路由协议，进行路由发现、路由维护等常见的路由操作，对接收到不是发向自己的分组要进行分组转发。移动自组网络的协议栈结构符合 OSI 分层模型，传输层以上为端到端协议，可以采用与有线网络相同的协议(如 TCP、UDP、FTP、HTTP 等协议)，因此链路层和网络层协议是移动自组网络协议研究的重点。

移动自组网络是为军事、航海、灾区等不依赖于固定通信基础设施的无线通信情况而提出的，网络节点的移动性使得网络拓扑会发生动态变化；移动节点大多采用电池供电，能量供应非常有限，所以节点发射功率不能很大，受节点无线发射功率的限制，网络中的节点与其发射功率覆盖范围之外的节点进行通信需要通过多跳路由实现。针对这些固有问题，移动自组网络设计的重点通常在网络层以及 MAC 层协议的设计上，MAC 协议是移动自组网络无线信道共享的基础，它保证各个节点公平地接入信道，共享网络

带宽资源；网络层协议包括拓扑控制和路由协议，是移动自组网络技术的核心。通过为移动自组网络设计高效的网络层以及 MAC 层协议，以达到网络在服务质量、能耗、安全性等方面的要求。另外，蜂窝网络因为障碍物遮挡或者基站之间的服务空洞造成某些区域无服务或者服务质量差的问题，这些问题也可以通过在蜂窝网络中引入多跳的移动自组网络来解决。

2．无线传感器网络

作为移动自组网络的一个子集，无线传感器网络（wireless sensor network，WSN）是一组传感器以自组织方式构成的无线网络，传感器节点协作感知、采集网络覆盖区域中感知对象的信息，并将感知信息发送给观察者。无线传感器网络可以部署在人员不方便到达的地方，用来检测这些地方的信息，例如，观测农林以及灾区环境，检测办公大楼内部的情况，为军队收集情报以及控制生产制造系统等。传感器网络通常包括传感器节点（sensor node）、汇聚节点（sink node）和管理节点。传感器节点分布在某一监测区域内，节点以自组织的形式构成网络，通过传感器节点间多跳中继方式将监测数据传送到汇聚节点，最后通过因特网或其他网络通信方式将监测信息传送到管理节点。同样地，用户可以通过管理节点进行配置管理，发布监测任务，告知传感器节点收集监测信息。

无线传感器网络结构如图 1-5 所示。其中，传感器节点是一个具有信息收集和处理能力的微型嵌入式系统，集成了传感器模块、信息处理模块、无线通信模块和能量供应模块。传感器模块负责监测区域内信息的采集和转换，信息处理模块负责管理整个传感器节点的操作、存储和处理自身采集的数据或者其他节点发送来的数据，无线通信模块负责与其他传感器节点进行通信，能量供应模块负责为整个传感器网络提供运行所需的能量。从网络功能上看，每个传感器节点兼顾传统网络节点的终端和路由器双重功能。目前传感器节点的软件硬件技术是传感器网络的研究重点。汇聚节点作为接收/发送节点，其数据处理、存储能力和通信能力相对比较强，它连接传感器网络和因特网等外部网络，实现两种协议栈之间的通信协议转换，同时把收集到的网络节点的监测信息发布到外部网络上。为了提供各种具体的应用支持，无线传感器网络的应用支撑技术包括了网络分层协议、节点定位和时间同步、分布式网络服务接口和分布式网络管理接口等。

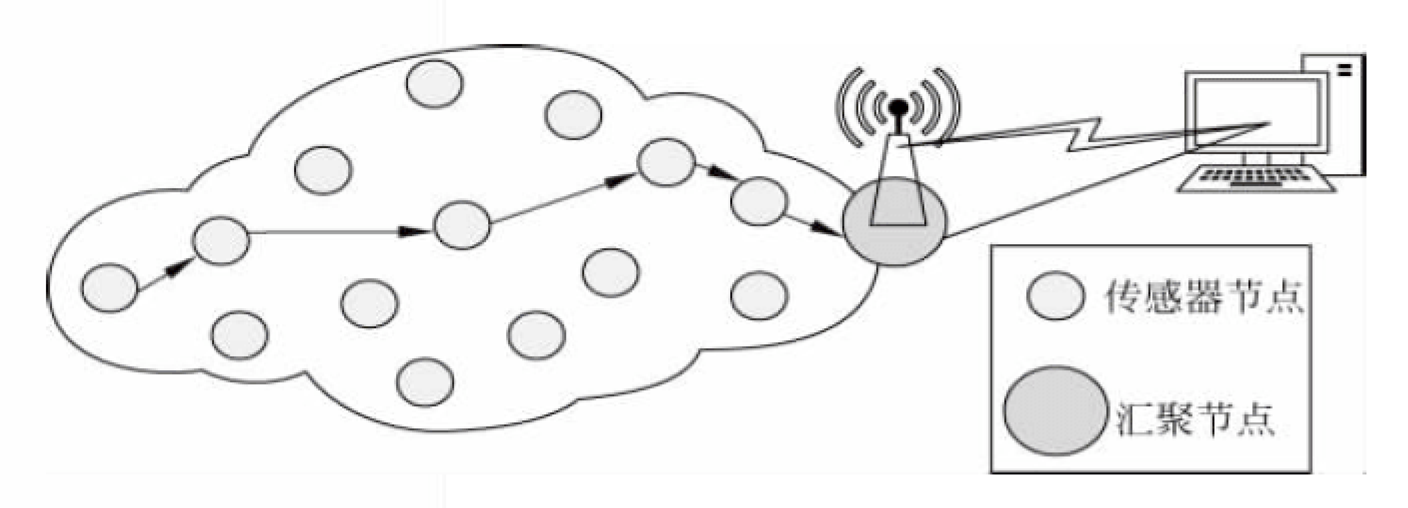


图 1-5　无线传感器网络结构

3．无线 Mesh 网络

无线 Mesh 网络（wireless mesh networks，WMNs）又被称为无线网状网或无线网格网，不同于传统点对多点网络结构，WMNs 是点对点的 Mesh（网状）网络结构。我们熟知

的因特网就是一个 Mesh 网络的典型例子。当发送一份 E-mail 时，电子邮件并不是直接到达收件人的信箱中，而是通过路由器从一个服务器转发到另外一个服务器，经过多次路由转发才到达用户的信箱。在转发过程中，路由器一般会选择效率最高的传输路径，以便使电子邮件能尽快到达用户的信箱。因此，无线 Mesh 网络可看作“Internet 的无线版”。WMNs 的核心是让每个节点既作为主机运行自己的应用程序又作为路由器转发其他节点的数据，整个网络通过自动配置实现节点间的互联，各用户节点可以通过相邻的其他用户节点，以多跳方式实现到骨干网的连接。

Mesh 结构可以用于不同种类的网络，国际标准化组织也积极考虑在各种无线网络标准中加入对 Mesh 组网方式的支持，如 IEEE 802. 16 无线城域网、IEEE 802. 11 无线局域网、IEEE 802. 15 无线个域网。IEEE 802. 16 标准组在 2003 年 4 月颁布的 IEEE 802. 16a 宽带无线城域网标准中设计了对点到多点和 Mesh 两种拓扑结构的支持。IEEE 802. 11 无线局域网工作组在 2004 年初也成立了 Mesh 研究组 802. 11s，主要研究支持无线分布系统的协议，以便实现无线局域网的多个 AP 之间通过自动配置拓扑的方式组网。IEEE 802. 15 无线个域网工作组在 2003 年 11 月成立了 Mesh 研究组 TG5，研究利用短距离、低成本设备通过 Mesh 方式来覆盖一个较大的环境，如家庭、学校、医院等。Wi-Mesh 联盟和 SSEMesh 组织是无线 Mesh 标准的主要推动者，Wi-Mesh 和 SSEMesh 提出的联合提议，已经于 2009 年形成了 IEEE 802. 11s 的标准草案（Draft D3. 00）。此外，ITU、3GPP 以及 IETF 等机构也在积极推动无线 Mesh 网络技术的研究、应用和推广。

WMNs 的网络实现模式可分为三类：基础设施型（wireless mesh backbones）网络配置模式、客户端型（wireless mesh clients）网络配置模式以及混合型网络配置模式。其中，混合型 WMNs 将基础设施型和客户端型两种网络配置模式综合在一起，综合了这两种实现模式的优点。混合型网络配置模式 WMNs 如图 1-6 所示，可以包括各种异构的无

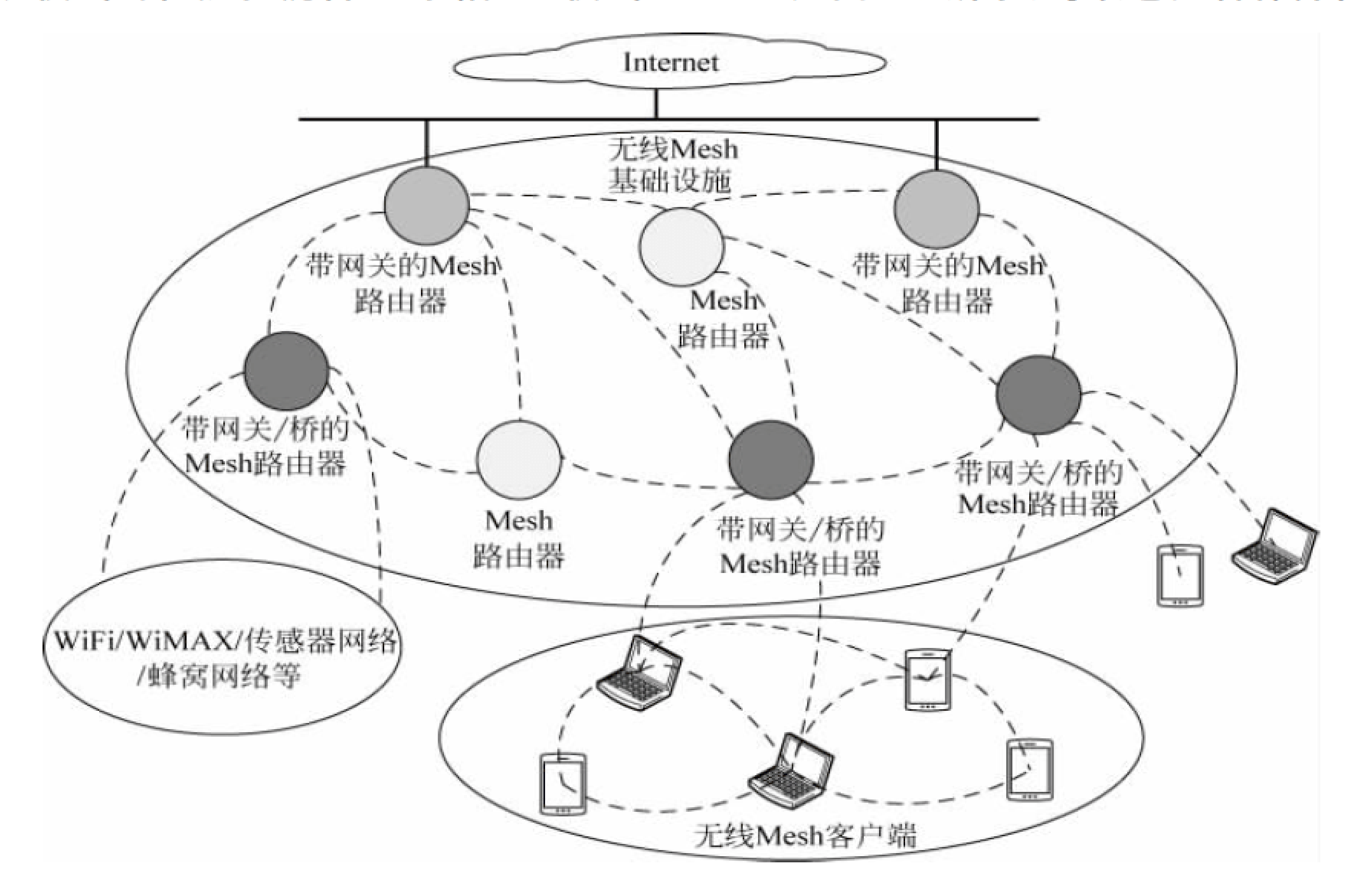


图 1-6　混合型 WMNs 网络结构

线与有线网络，如无线局域网、WiMAX、蜂窝网络、移动 Ad hoc 网络、无线传感器网络以及因特网等。著名的“无线台北”就是典型的混合结构的无线 Mesh 网络。类似的项目还有美国的费城、芝加哥、旧金山、匹兹堡等，新加坡也实现了 WiFi Mesh 全境覆盖。

对 WMNs 的研究已经遍及网络各个层面，其中包括应用层协议、传输层协议、网络层协议、MAC 层协议、物理层传输技术，以及跨层设计方法、网络管理机制、安全解决方案等。WMNs 作为 Internet 的重要扩展，所使用的传输层协议需要与 Internet 中广泛使用的各种协议兼容。对于网络层的协议主要集中在路由机制上，由于 WMNs 与 Ad hoc 网络结构的相似性，部分 Ad hoc 网络中的路由协议也可以应用于 WMNs，而为了提供高质量 Internet 接入服务，在设计路由协议时则需要考虑链路质量。由于链路传输数据时会由于共享信道等原因对邻近的链路产生影响，因此 WMNs 的 MAC 层协议较之 WLAN 等单跳网络存在一些新的问题，MAC 层协议的设计也受到了广泛的关注。为了提供高速率、高吞吐率的 Internet 接入服务，WMNs 需要不断引入最新的物理层传输技术，目前主要是 MIMO 和 OFDM 技术。WMNs 的网络管理包括了移动性管理、能量管理以及网络监控。WMNs 的网络安全研究则主要是设计安全的路由、MAC 协议以及开发安全监控响应系统。

1.2 有线网络的发展

从 1897 年马可尼成功演示无线电报，1901 年实现了横跨大西洋的无线电接收，之后相当一段时间无线电通信技术快速发展，至今已经有一百多年的历史。在这一百多年中，涌现出了多种类型的无线系统，而后又不断消失，例如早期的电视传输是通过无线电发射机实现广播的，由于无线电传输的电视节目信号较差，后来逐渐被有线电缆传输所取代。类似地，构成电话网骨干的点对点微波线路正逐渐被光纤取代。在第一个例子中，安装分布式有线电视网络后无线技术就变得过时了；而在第二个例子中，新的有线技术（光纤）取代了旧的无线技术。相反的例子是在如今的电话通信领域中，无线（蜂窝）技术在某种程度上取代了有线电话网络（在部分地区有线网络并没有得到很好的发展）。这些实例表明，在许多情况下都存在无线技术与有线技术间的选择，并且这种选择通常随着新技术的出现而变化[1]。下面介绍有线网络的发展历程与关键技术。

1.2.1 互联网

1. 互联网的起源与发展

20 世纪 60 年代起源于美国的因特网现已发展成为世界上最大的国际性计算机互联网，当时电话网是世界上占据统治地位的通信网络。而随着计算机变得越来越重要，人们很自然地想到将分布在不同地理位置的多台计算机连接起来，方便各台计算机互相分享信息数据。电话网使用电路交换的方式使任意两部电话通信，对于在一段时间内持续发出语音信息的话音通话业务来说，在通话期间独占一条专用物理通路的电路交换方式是合适的选择。然而与电话网中话音业务的特点不同，计算机产生的数据往往是突发性

地出现在传输线路上的，因此线路上真正用来传输数据的时间往往不到 10% 甚至低于 1%，计算机用户之间若采用电路交换的方式通信，所占用的通信线路资源在绝大部分时间内都将是空闲的。因此，美国多个高校的研究者开始研发不同于且可以替换电路交换的、适用于计算机通信的新的交换方式，最终由美国国防部高级研究计划署(advanced research projects agency，ARPA)于 1969 年 11 月，建立了一个名为 ARPANet 的网络，首次采用分组交换的方式将分布在洛杉矶的加州大学洛杉矶分校、加州大学圣巴巴拉分校、斯坦福大学、犹他州大学 4 所大学的 4 台大型计算机连接起来。ARPANet 便成为世界上第一个分组交换网并成为今天公共互联网的前身。到 1972 年，ARPANet 已经扩大到具有 15 个主机节点，并完成了第一个 ARPANet 终端系统之间主机到主机协议，也就是网络控制协议(network-control protocol，NCP)。利用这个终端到终端的协议，便可以设计应用程序，1972 年 Ray Tomlinson 写了第一个电子邮件程序。

ARPANet 是单个的、封闭的网络，想要连接到 ARPANet 上的主机，都需要与就近的节点交换机相连。然而，在 20 世纪 70 年代中期，涌现出其他多个单个的分组交换网，例如，ALOHANet，一个连接夏威夷岛多所高校的微波网络；Cyclades，一个法国的分组交换网。这样单个独立的分组交换网越来越多，需要一个网络将这些各自独立的网络连接起来，于是美国国防部高级研究计划署(defense advanced research projects agency，DARPA)开始推动这项工作，旨在建立网络的网络[5]，即“互联网”。1983 年，TCP/IP 成为 ARPANet 上的标准协议，使得所有使用 TCP/IP 的计算机都能利用互联网通信，因而人们把 1983 年作为因特网诞生的时间。从 1985 年起，美国国家科学基金会(national science foundation，NSF)就围绕 6 个大型计算机中心建设计算机网络，即国家科学基金网 NSFNET。它是一个三级计算机网络，分为主干网、地区网和校园网(或企业网)。这种三级计算机网络覆盖了美国主要的大学和研究所，并成为因特网中主要的组成部分。1991 年，NSF 和美国的政府机构开始认识到，因特网不应仅限于大学和研究机构，应该扩大其范围，于是世界上许多公司纷纷接入因特网。网络上的通信量急剧增大，现有的因特网容量满足不了需要，于是美国政府决定将因特网主干网交给私人公司来经营，并开始对接入因特网的单位收费。

从 1993 年开始，由美国政府资助的 NSFNET 逐渐被若干个商用的因特网主干网替代，政府不再负责因特网的运营，这就出现了新的名词：因特网服务提供者(Internet service provider，ISP)。在许多情况下，因特网服务提供者就是一个进行商业活动的公司，因此 ISP 又常译为因特网服务提供商，例如我国的中国电信、中国联通和中国移动。ISP 可以从因特网管理机构申请到很多 IP 地址，同时建造通信线路，铺设路由器等连网设备，任何机构和个人通过向某个 ISP 缴纳规定的费用，就可以从该 ISP 获取 IP 地址使用权，并可以通过该 ISP 接入因特网。

2. 计算机网络体系结构

由于互联网产生之初始是多个各自独立的网络，并且所用的计算机也来自不同厂家，设备生产厂商为了能够垄断市场，也各自制定自己的体系，对于其他厂家生产的设备不能兼容联通，从而无法交互。这使互联网成为一个非常复杂的系统，设想一下，连接在互联网上的可能来自于不同的网络体系的两台计算机，互相交换信息。除了需要有一条

传送数据的通路，这两台计算机之间还有很多工作需要完成，例如，发起通信的计算机必须将通信的通路进行激活，也就是发出一些信令，保证要传送的计算机数据能在这条通路上正确地发送和接收；要告诉网络如何识别接收数据的计算机；若计算机的文件格式不兼容，则至少有一台计算机能够完成格式转换；对于传输过程中有可能出现的事故，如数据传送错误、重复或丢失、网络中某个节点交换机出现故障，应当有可靠的措施保证对方计算机最终能够收到正确的文件。由此可见，两个计算机系统想要有条不紊地交换数据、相互通信，必须高度协调工作，建立一种大家都必须遵守的标准，这样才能让不同的计算机按照一定的规则进行"谈判"，并且在谈判之后能"握手"，也就是必须遵守一些事先约定好的规则。

为了使不同体系的计算机网络都能畅通无阻地互联，国际标准化组织(ISO)和国际电报电话咨询委员会(CCITT)联合制定了开放系统互连参考模型(open system interconnect reference model，OSI/RM)。"开放"是指一个系统只要遵循 ISO 标准就可以和世界上任何一个也遵循这一标准的系统进行通信。1983 年形成了开放系统互连基本参考模型的正式文件，即有名的 ISO 7498 国际标准[6]，也就是 7 层协议标准，7 层从低到高依次是物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。然而，OSI 只是提供了一种功能结构的框架，由于其过于复杂甚至层次划分并不十分合理等原因，导致在市场化方面 OSI 没能成功推行。真正占领市场的标准是 TCP/IP，该标准体系结构与 OSI 相比相对比较简单，只有四层：应用层、运输层、网际层和网络接口层。两台计算机在通信过程中，应用进程的数据在各层之间传递，简单起见，假设两台计算机之间经由一台路由器，两台计算机之间数据传递过程如图 1-7 所示。

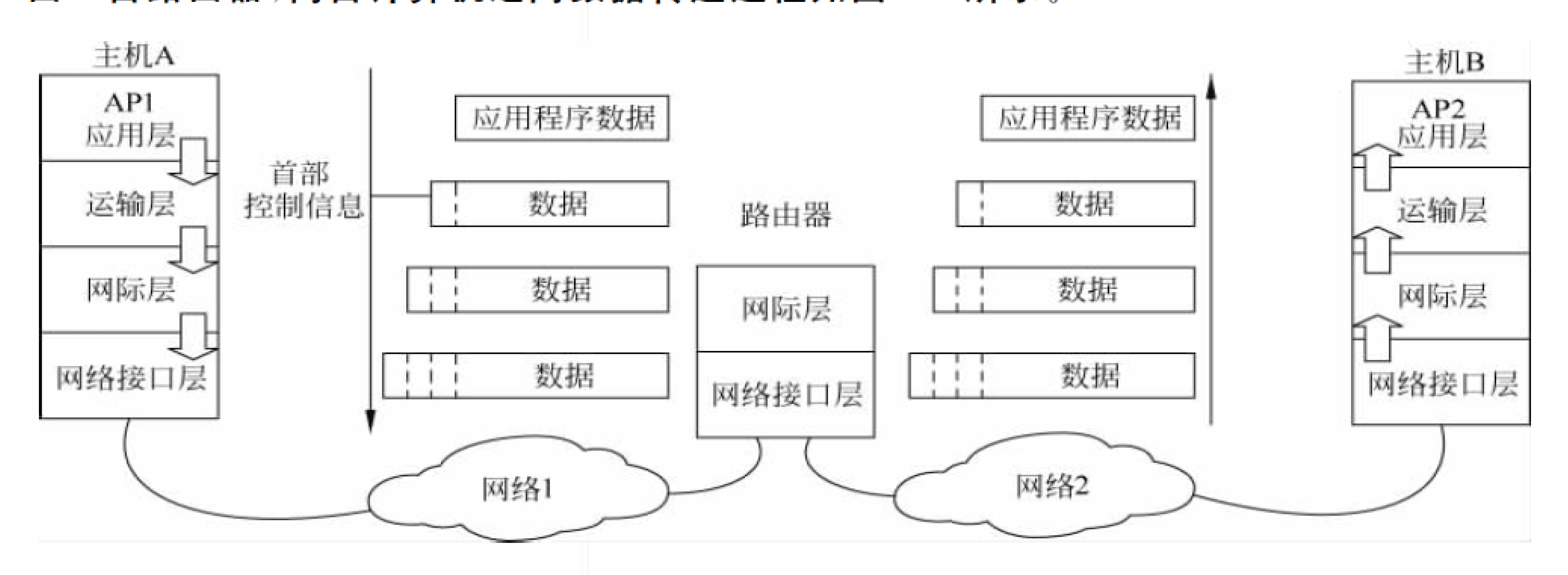


图 1-7　计算机网络体系结构

主机 A 的应用进程 1 的数据由最高层应用层加入本层控制信息后，依次向下一层传递并依次加入该层控制信息，最终以比特流的形式通过底层物理设备传送至目的地，即主机 B。主机 B 在底层接收到数据比特流，拆除本层控制信息后，依次传送至上层并拆除该层的控制信息，最终主机 B 收到主机 A 的数据。这个过程对于用户来说，都被屏蔽掉了，主机 A 应用进程 1 感受的是直接将数据交给了主机 B 的应用进程 2。对于每一层来说，也同样感觉直接将数据交给了它的对等层。各层之间的协议，实际上就是在各个对等层之间传递数据时的各项规定。

1.2.2 有线电视网络

1. 有线电视网络发展历程

早期的电视传输是通过无线电发射机实现广播的，后来逐渐被有线电缆传输所取代。1948年，在美国宾夕法尼亚州位于山谷的曼哈尼城，由于大多数居民居住在当地三个电视台的阴影区，电视信号被阻挡，收看电视的效果极差。有一位专营电视装置的约翰·华生想到在山边最高处安装增益较高、性能良好的电视接收天线，征得电力公司的许可后，架设了一些同轴电缆，将天线接收信号分送给阴影区的每家用户，这就是世界上最早的公共天线系统（master aerial television，MATV）。20世纪50年代初，这种收视系统被移植到了城市里，有效解决了接收重影严重影响收视质量、开路发射天线的零点信号微弱无法正常接收、天线林立影响市容等城市中开路电视个体接收所存在的若干问题[7]。当时的系统已经拥有了非常简单的前端，可以实现少数几个开路电视频道的直接混合和信号的有效分配，这便是早期的共用天线系统（community antenna television，CATV）。为了更充分地发挥系统的作用，并满足用户不断增长的对电视节目丰富度的需求，CATV系统中增加了自办节目频道，而自办节目的出现又促进了CATV系统进一步扩大范围。于是在整个五六十年代，丰富节目套数与扩大系统覆盖范围相辅相成共同促进了CATV系统技术的发展。20世纪70年代初，通信卫星传送电视信号进入实用阶段，利用CATV系统实现卫星电视的共同接收便成为一个切实可行的方案。自此之后，CATV的功能有了很大提升，传送的电视节目的套数也得到极大丰富，CATV系统的覆盖范围也进一步扩大，并逐渐发展为今天的有线电视系统（或称电缆电视系统，cable television）。随着光纤通信技术的发展，CATV的传输方式也从同轴电缆发展到光纤传输。20世纪80年代末期至今，随着因特网的迅猛发展，社会信息化程度提高，人们对信息服务需求的增长，有线电视CATV也在向综合信息服务网发展，使有线电视网络逐渐具备双向传输与交互功能。

有线电视系统，无论其规模大小与繁简度如何，均可抽象为以下4个组成部分：信号源、前端、干线传输系统、用户分配网，如图1-8所示。信号源一般是指有线电视的节目来源，包括卫星转发的卫星电视信号、当地电视台发送的开路电视信号、自办电视节目、当地微波台发射的微波电视信号以及其他有线电视网传输过来的电视信号等，接收或产生这些节目信号的设备共同组成了系统的信号源部分。前端是位于信号源与干线传输系统之间的设备组合，是系统的信号处理中心，任务是把从信号源送来的电视节目信号进行滤波、变频、放大、调制、混合等，使其适合于在干线传输系统中进行传输。干线传输系统将前端输出的高频复合电视信号优质、稳定地远距离传输给用户分配网，传输方式主要有光纤、微波、同轴电缆。最终，有线电视信号经由用户分配网，被准确高效地分送到千家万户，用户分配网一般由分配放大器、延长放大器、分配器、分支器、用户终端盒（也

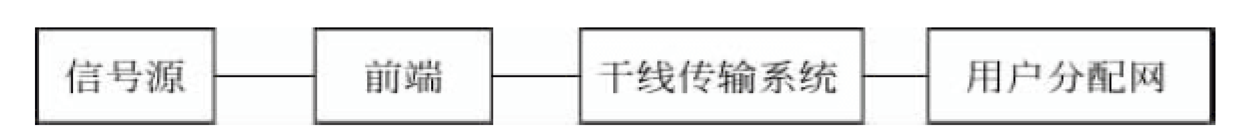


图1-8 有线电视系统的物理模型

称系统输出口)以及连接它们的分支线、用户线等组成。

作为信息高速公路的一部分,有线电视系统逐渐发展为数字化、综合化、网络化以及智能化电视网络。数字电视是数字化信息技术的产物,是将传统的模拟电视信号经过量化编码为数字式电视信号,使电视信号更加易于处理与传输。有线电视网络的服务内容也在不断丰富,逐渐建立起集数据、语音、视频图像为一体的宽带多媒体综合业务平台。相较于传统广播电视业务由单个独立的有线电视系统独自实现,卫星的存在已将一个个孤立的有线电视系统连成一个有线电视网络体系,从而为整个有线电视网络覆盖下的用户提供更为丰富的电视节目。有线电视网络的智能化则是伴随着数字化、综合化与网络化的进程中,逐渐告别过去传统的单一服务模式,实现双向交互,发展为现代综合信息服务网。

2. 现代有线电视网络

现代有线电视网络所提供的业务,既有基本业务,又有增值业务和扩展业务;传送的信号类型,既有模拟电视信号,又有数字电视信号和 IP 数据信号。为了实现多种综合业务,有线电视系统不再是自成一体的独立系统,而是通过上一级的数字光纤骨干环网和本地光纤骨干环网实现与其他有线电视系统的联网,另外,与公共电信网也可互联。本地的传输覆盖网则采用混合光纤同轴电缆(hybrid fiber coaxial,HFC)模式构成双向传输分配网,其中光纤传输部分采用空间分割(空分复用)的方式,即上、下行的信号分别用不同的光纤传输。现在的有线电视网络在组成上比传统有线电视系统复杂得多,数字电视信号源主要是数字卫星电视 TS 流、视频服务器和业务生成系统等;数字电视前端实际上是一个数字电视多媒体平台,包括复用器、条件接收系统、数字调制器等,数据前端则主要是电缆调制解调器(cable modem)的前端控制器(cable modem termination systems,CMTS)。现代网络与传统网络的不同之处,除了数字模拟并存、信息交互双向传输以及互联互通之外,最重要的不同在于,有线电视综合业务的实现要求现代有线电视网络具有复杂完善的管理控制系统,包括用户管理与授权,系统、网络、设备管理,条件接收、节目播出、媒体资源、收费管理等一系列确保有线电视网络为广大电视用户提供稳定优质服务的管理控制系统。有线电视网中的用户安装机顶盒就能够收看数字电视,并获得授权享受个性化的视频服务和其他增值服务。此外,加电缆调制解调器就可实现与计算机进行通信、接入因特网、提供 IP 电话等功能。目前,世界上应用比较广泛的数字电视标准制式主要有欧洲的数字视频广播标准(digital video broadcasting,DVB)以及北美洲的数字电视国家标准(advanced television systems committee,ATSC)。我国使用的是中国移动多媒体广播标准(china mobile multimedia broadcasting,CMMB)。

1.2.3 光网络

从烽火到电报,再到 1940 年第一条同轴电缆正式建成用于通信,这些通信系统的性能与容量不断提升,但是却各有其极限:使用电信号传递信息虽然快速,但是传输距离会因为电信号容易衰减而需要大量的中继器;微波通信虽然可以使用空气作为传输介质,可是也会受到载波频率的限制。到了 20 世纪中叶,人们才渐渐了解到使用光来传递信息,能带来很多过去的通信系统所没有的显著好处。然而,当时并没有同调性高的发光

源，也没有适合作为传递光信号的介质，所以光通信一直停留在概念层面。直到20世纪60年代，激光的发明才解决了第一项难题。1970年，美国康宁玻璃公司研制出损耗为20dB/km的石英光纤，美国和日本同时实现了半导体激光器在室温下连续工作，光纤作为通信的传输媒介同时具有容易调制的光源，这为光纤通信的发展与应用拉开了序幕。1976年，速率为44.7Mb/s的第一个光纤通信系统在亚特兰大顺利进行了现场试验，该系统经过全面性能测试后，很快商用。20世纪80年代，光纤通信以其巨大的带宽、很低的损耗在通信网络中获得广泛应用，并逐渐替代了传统的电缆通信。

20世纪末，是光通信和光网络发展最为繁荣的几十年。光通信经历了从低速到高速、从准同步数字序列（plesiochronous digital hierarchy，PDH）到同步数字序列（synchronous digital hierarchy/synchronous optical network，SDH/SONET）再到光传送网（optical transport network，OTN）的发展历程。20世纪80年代末，同步数字序列（SDH/SONET）问世，在历史上第一次实现了全球统一的传送网标准，规范了光接口，而且定义了对光信号质量的监控、故障定位和远程配置等重要的网络管理功能。然而，SDH主要在光域起传输介质的作用，信息处理仍然在电域完成，传统的电网络无法直接在光层进行多工、切换以及路由等工作，网络交换节点需要使用光电转换设备将光信号转换为电信号再将电信号转回光信号，总体传输速率因使用光电转换设备而受到限制。波分复用（wavelength division multiplexing，WDM）技术进一步挖掘了光纤的带宽潜力，扩充了光纤的传输容量，同时也为光层联网提供了可能。经过各国研究人员大量的理论与实践研究，20世纪末，国际电信联盟标准化部（ITU-T）提出光传送网（OTN）的概念，提出以波长作为交换粒度，通过光交叉连接设备（optical cross connection，OXC）和光分插复用设备（optical add-drop multiplexer，OADM）实现组网。进入21世纪后，WDM提供的带宽已满足当前通信流量的需求，服务质量保证以及人们对网络的智能化和自动化需求不断提高，这些促进了光网络向智能化方向发展，自动交换光网络（automatically switched optical network，ASON）便产生了。

ASON代表智能光网络的主流方向[8]，无论技术研究还是标准化进程都进展迅速，2005年左右推出商用。ITU-T先后制定了G.807、G.8080以及后续的ASON相关标准，IETF、OIF等组织也积极扩展MPLS协议，使其能够成为ASON的路由和信令协议。ASON的核心特点是将网络的控制功能和管理功能分离，电子交换设备动态向光网络申请资源，ASON根据网络中业务分布模式动态变化的需求，通过控制平面的路由和信令机制实现邻居和业务的自动发现，以及连接的自动建立和删除，不需要人工干预。管理平面包含对网络资源、控制平面以及传送平面等内容的管理，管理平面与控制平面相互补充，实现光网络资源灵活高效配置、性能监测以及故障管理等功能。传送平面则是数据传送实体的集合，即光网络中的物理设备。自动交换光网络技术使原来复杂的多层网络结构可以变得简单和扁平化。

1.3 异构网络融合

目前，用于通信的网络既有无线网络也有有线网络，无线网络包括宽带无线接入网、蜂窝移动网以及卫星网络等不同种类的网络。各种异构网络相互交叠，如图1-9所示，共

同满足用户无缝接入的需求并提供多种多样的服务。由于特定网络通常只提供特定类型的服务,这使得用户在需要不同种类的服务时,需要在不同的网络之间切换,而一个网络只用来提供特定的服务也是对网络资源的浪费。为了充分利用现有网络资源为用户提供高质量服务,网络融合成为全球的信息与通信领域数十年来孜孜不倦追求的理想目标。国际通行的网络融合内涵,是指从分离的网络、分离的业务演进到统一的网络来提供各种综合的业务服务[9]。

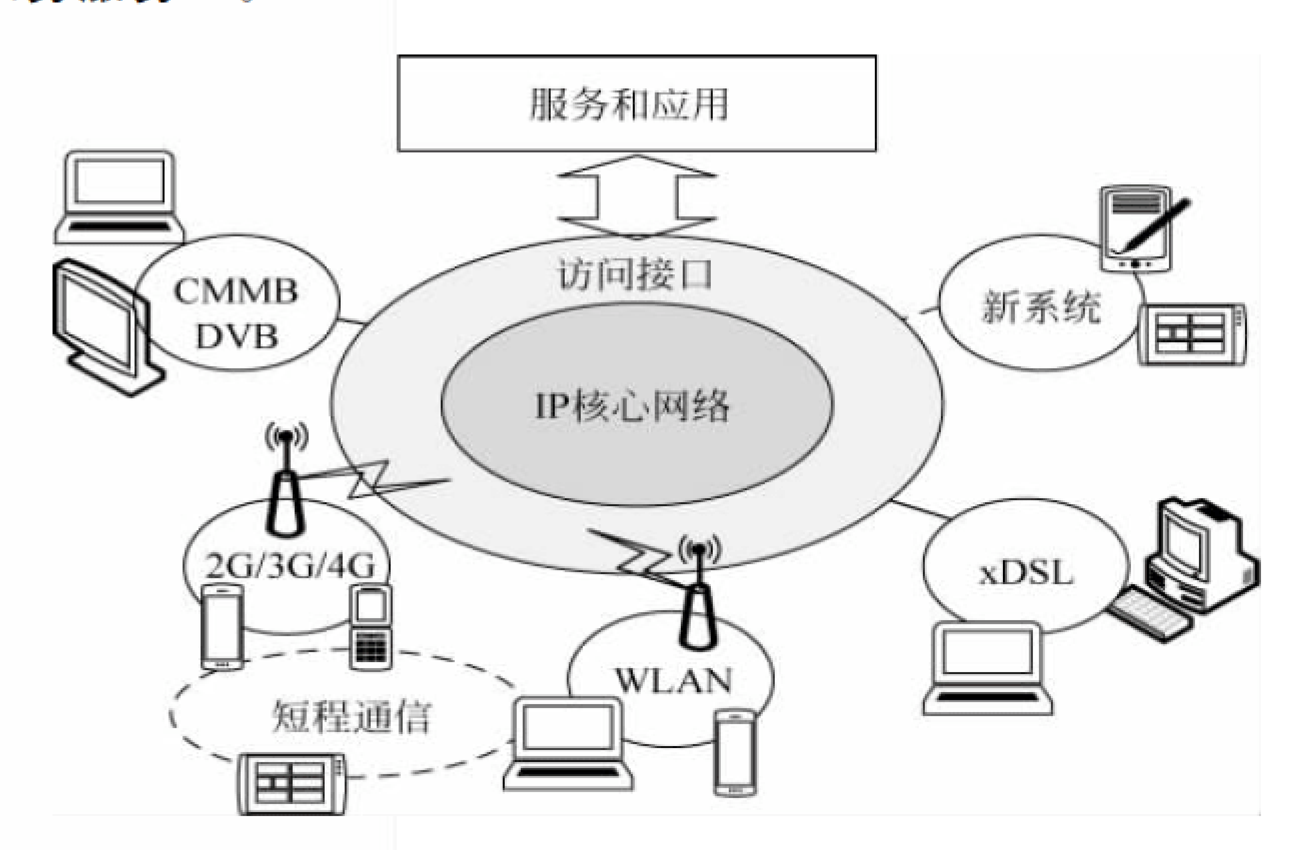


图 1-9 异构网络体系结构

网络融合应当以用户需求为导向,给用户提供高质量、丰富、便捷的服务;同时,其动力应该是充分利用现有的各种网络资源,降低网络的建设和运维成本,增加运营商利润,而不是全业务牌照发放,即不能简单地理解为各个不同网络运营商获得全业务的运营许可,否则会造成网络严重的重复建设,资源严重浪费。目前各种现有网络并没有得到充分融合,制约了网络资源和服务的共享与利用。网络融合从技术上来说涉及承载网络、业务控制、接入网、驻地网、固网与移动网络融合、终端、应用等各个层面,是业界长期追求的目标,也将会是长期的过程。

1.3.1 异构无线网络融合

近几十年来无线通信技术迅猛发展。蜂窝移动通信技术更新换代,从 2G 话音与低速数据服务到现在 4G 高速数据传输,不断向移动宽带化发展;以 IEEE 802.11 系列标准为代表的无线局域网从最初的宽带无线接入,逐渐向支持用户移动的宽带移动化方向发展;同时,卫星网络、数字电视广播则为用户提供更丰富的服务内容。各种无线网络的组网模式、网络结构、技术参数、控制与管理机制、业务类型均有所不同,并且可能属于不同的所有者(如运营商、企业专网、个人网络等)。无线网络异构性特征越来越明显,多种无线通信网络并存,共同为用户提供不同种类的服务,满足不同用户的各种需求。没有哪一个网络能够独自为用户提供所有服务并满足所有要求,异构无线网络之间的互通与融合已经成为无线网络发展的必然趋势。

ITU、ETIS、IETF 以及 3GPP 等标准化组织都在致力于无线网络融合技术的规范化

与标准化工作。根据3GPP的描述，网络融合是采用通用、开放的技术实现不同网络或网络元素的合并或者融合；根据ITU的定义，网络融合就是通过互联、互操作的电信网、计算机网和广播电视网等网络资源的无缝融合，构成一个具有统一接入和应用平台的高效网络，使人类能在任何时间和地点，以一种可以接受的费用和服务质量，安全地享受多种方式的信息应用。从目标上来看，融合技术旨在异构网络环境中提供统一的业务；从技术本质来看，融合方案均在不同层面上屏蔽异构性[10]。异构无线网络在融合过程中涉及移动性管理、无线资源管理、服务质量以及网络安全性等问题。

(1) 移动性管理。移动性管理技术，尤其是切换技术，是实现异构无线网络融合的关键技术之一。过去对切换管理技术的研究多集中在同构网络中，也就是单一的通信系统内，切换管理的主要目标是保证语音业务或者数据业务的连续性和可靠性。而异构网络情况下，各种网络的空中接口技术及相关协议具有差异性和不可兼容性，需要综合考虑多种异构网络参数从网络的角度重新设计切换管理方案，从而使用户可以在网络之间无缝切换。

(2) 无线资源管理。除了传统网络无线资源管理中需要考虑的频谱、码字、时间以及功率等资源，异构无线资源还包括移动用户的接入权限、连接模式等与异构网络结构相关的资源。这就使异构融合网络中的资源管理方式与传统无线资源管理方式有所不同。异构融合网络资源管理需要考虑所有各种网络的管理机制，通过应用多种接入技术、可重配置或者多模终端技术、支持智能的呼叫和会话接纳控制技术以及业务和功率的分布式管理技术，实现对各种网络中无线资源的最优化利用。

(3) 服务质量。不同类型的业务对网络性能的要求千差万别，以业务需求为中心，协调调用各种网络资源，为端到端业务提供良好的服务质量，是异构融合网络需要面对的问题。端到端的业务提供是在业务层、网络层、终端以及管理系统等多个层面融合的基础上进行的。因此，在异构融合网络中对服务质量的保障要以网络间的信息互通作为基础，既需要从高层进行宏观的运营管理，又需要从底层进行微观的网络资源控制，从而实现网络资源对业务的保障具有连续性。

(4) 安全融合。同所有的网络一样，安全问题同样是无线异构融合网络需要面对的一个重要问题。由于不同网络的体系结构和安全威胁的差异，各自的安全解决方案也存在很大差异。异构融合网络包含了原有各自网络所固有的安全问题以及由于网络融合而带来的新的安全问题，如网络之间安全协议的无缝衔接、网络间密钥协商的复杂管理和密钥负荷以及异构网络间统一接入认证问题等。针对不同的网络以及不同的融合方式的安全接入与管理机制是异构无线网络融合的一个关键问题，已经得到标准化组织以及学界的广泛研究。

目前比较成熟的融合方案有蜂窝网络与WLAN的融合，以及异构网络之间的切换技术。

(1) 蜂窝网络与WLAN融合。蜂窝网络与无线局域网是目前应用最为广泛的无线网络，二者间的融合是异构网络融合中的重点，受到业界与学术界的广泛关注。蜂窝网络与无线接入网络融合的典型例子是3GPP制定的3GPP网络和WLAN的互联融合方案。3GPP定义了基于IMS(IP multimedia subsystem)的网络之间互通，在R6版本中规定了基于IMS的3GPP网络与外部网络互联的相关标准，包括了需求规范、网络架构、安

全相关的规范以及流程相关的规范等。3GPP 在融合协议 TS22.934 中建议了六种 3GPP 网络与 WLAN 融合的互操作情景模式，按照 WLAN 与 3GPP 网络结合的紧密程度，分为松耦合和紧耦合两大类。紧耦合是蜂窝网络与 WLAN 共享核心网，在核心网内实现异构无线网络的融合；而松耦合是 WLAN 和蜂窝网络核心网在用户平面分别接入互联网，仅在控制平面共享认证、授权与计费功能。

(2) 异构网络之间切换。IEEE 802.21 工作组研究了如何在异种接入技术之间提供独立于媒体的切换能力(media independent handover，MIH)，其中定义的切换包括 IEEE 系列接入技术之间的切换以及 IEEE 系列和蜂窝网络之间的切换。融合网络安全性方面，UMTS 安全体系的关键组件是认证和密钥协商(authentication and key agreement，AKA)，而 WLAN 则有 IEEE 802.11i 和 WAPI 两种不同的安全体系。任何单一的安全体系都无法为融合网络提供安全保证。3GPP 在 TS33.234 定义了 3G 与 WLAN 融合网络的安全体系结构，使用 AAA 和 EAP(extensible authentication protocol)技术作为 3GPP 和 WLAN 之间的纽带，采用 EAP AKA 使得 UMTS AKA 可以经由 WLAN 在用户终端和 3GPP 系统之间执行，实现了统一用户身份管理。对于 3G 与 WLAN 的融合已经运用在 4G 网络中，然而，世界范围内完全实现无处不在、无所不能的异构融合网络，在技术上仍然还有很长的路要走。

1.3.2 三网融合

自从莫尔斯电码、贝尔电话的出现以及相关技术的不断革新，人类信息传播方式不断进步并发生了巨大的变化。21 世纪的一个重要的特征就是信息化，它是一个以网络为核心的信息时代。要实现信息化就必须依靠完善的网络，因为网络可以非常迅速地传递信息。因此网络现在已经成为信息社会的命脉和发展知识经济的基础。这里所说的网络包括电信网络、有线电视网络和计算机网络。三个网络建设之初的目的以及各自提供的服务均不同。电信网络为用户提供电话、电报以及传真的服务。有线电视网络为用户提供各种电视节目服务。计算机网络则可以使用户利用计算机强大的数据处理能力相互传送数据分享各种有用的资料，包括文字、图像以及视频等文件。三个网络在信息化的过程中各自起到了不同的重要作用，然而随着数字技术和光纤技术的发展，电信网、有线电视网、互联网趋向于相同的 IP 网络技术和结构，三个网络的边界越来越模糊，它进一步要求网络对其终端资源进行整合。在此背景下，传统电信运营商、有线电视网络运营商以及互联网运营商均可利用自身的网络交叉进入相互的市场，例如，用户可以通过电话接入互联网、计算机网络可以提供网络电视与网络视频电话等服务、可以通过数字机顶盒接入互联网。

“三网融合”这一术语在国内外学术界、标准化组织以及信息产业界都没有严格的定义。国际上有两个概念与“三网融合”类似：convergence(融合)或 tripleplay(三重业务捆绑)。从学术角度看，融合并不单指三网融合，涉及的领域比较宽泛。“三重业务捆绑”比较准确地概括了三类业务应用(语音、视频和数据)的融合。传统电信运营商、有线电视网络运营商以及互联网运营商通过三重业务捆绑，不再以产业予以冠名，取而代之的是经营着多种业务甚至全部业务、拥有着相似商业模式的网络运营商。

三网融合的过程并不是一蹴而就的，融合过程不仅涉及技术问题，还有各个融合主体的利益问题以及政策规制问题，如果不能很好地解决这些问题，三网融合将不容易有突破性进展。三网融合在政策法规上主要体现在融合立法和成立融合监管部门，通过立法取消政策性产业壁垒，打破行业垄断，允许在业务上相互竞争，随着各国政府相关政策的相继出台，三网融合的时代已经到来。在美国、英国、日本等国家，20 世纪 90 年代以来相继解除了电信企业与广电企业的相互进入的限制，电信与广电业相互竞争，均可提供对方的服务，在广电企业和电信企业的竞争合作的基础上，融合服务市场得以迅速发展。

中国的三网融合相关政策经历了从严格管制到逐渐放松的过程。严格管制时期，曾有国务院办公厅[1999]82 号文“电信不得从事广电业务，广电不得从事通信业务”的规定。2008 年 1 月，国务院发布[2008]1 号文《国务院办公厅转发发展改革委等部门关于鼓励数字电视产业若干政策的通知》，解除了对网络融合的政策限制。2009 年 5 月，国家发展和改革委员会《关于 2009 年深化经济体制改革工作的意见》中首次提出“实现广电和电信企业的双向进入”。2010 年 6 月，国务院办公厅印发第一批三网融合试点地区（城市）名单。随着时间表的推出和首批三网融合试点地区的确定，中国三网融合的发展前景明朗，然而在技术、政策监管等方面仍然存在问题。技术方面，与美国、欧洲、日本等国家的网络运营商相比，中国的电信业与广电业的宽带网络建设和运营均无法满足融合业务要求。监管方面，多方监管问题依然存在，产业间各自为政，产业隔阂严重。因此，中国三网融合服务并未广泛普及。电信和广电业需要进一步推进技术发展以及开放竞争，以加快三网融合的步伐。

1.3.3 终端融合

随着无线通信技术的迅猛发展和人们对移动接入的需求不断提升，各种无线接入技术纷纷涌现。无线通信网中多种不同标准的异构网络并存：用于蜂窝覆盖的 GSM、WCDMA、TDD-LTE-Advanced 等制式不同的几代移动通信网，各种宽带无线接入网络，用于超大范围覆盖的卫星通信等。各国已经开始研究 5G 网络，未来无线通信网络将长期存在多重标准、多种制式。与此同时，用户所拥有的接入网络的终端设备也日益丰富，包括智能手机、便携式电脑以及消费类电子设备等。

为了使用户充分享受异构网络带来的方便、快捷的接入体验，集成多种网络接口的多模终端日益普及。多模终端如图 1-10 所示，是指在终端增加通信模块支持的网络制式种类，能够支持多于一种无线接入技术，并能够通过这些无线接入技术系统获得服务的

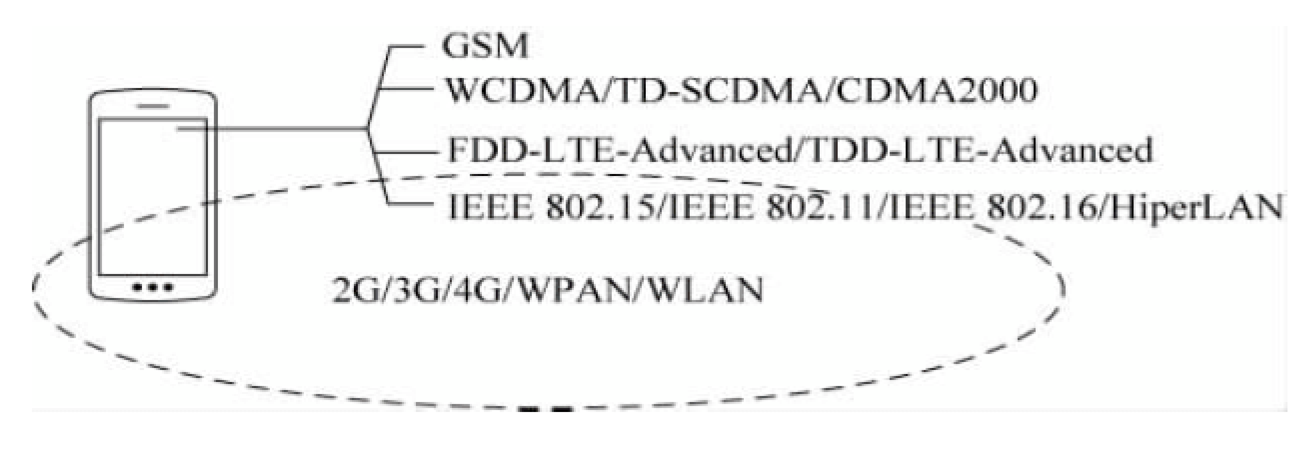


图 1-10　多模终端

终端。由于无线接入技术种类繁多，3GPP采用多模终端模式的工作方式对多模终端进行定义分类：手动模式变换多模单待终端、自动模式变换多模单待终端、双收单发多模双待终端以及双收双发多模双待终端等。具体采用哪种多模方式主要取决于终端使用的方便性以及市场需求等因素。

多模终端的使用使得以下服务成为可能：当终端上有多种不同类型的业务流需要传输时，可以根据不同类型业务流的特点、不同网络的当前链路状况、用户的喜好等因素把不同类型的业务流分发到不同的网络接口上进行传输，在保障各种业务服务质量的基础上，降低用户的通信成本和能量损耗。当某个接入网络不再可用或服务质量明显下降时，可以及时作出反应，将此网络上承载的业务流无缝地切换至其他可用网络。为了达到这样无缝接入、切换，数据高效传输的用户体验，多模终端的设计过程中需要考虑一系列技术指标：

（1）初始搜索时延。多模终端支持的模式、频段较多，在有联网需求时要对网络服务范围内的信号进行遍历搜索，搜索模式决定了终端完成小区驻留的时间。

（2）协议栈复杂度。多模终端需要在一块基带芯片集成多种模式的协议栈，导致芯片实现复杂度大大增加。

（3）模式之间干扰。2G/3G/4G/WLAN多模终端在两种模式并行收发数据时，存在一个终端模式间的干扰问题。

（4）终端功耗问题。网络支持的数据传输速率不断提升，提供的业务类型不断丰富，对于多模终端来说，优越的服务必然带来更大的功耗。

1.3.4 固定移动融合

固定移动融合（fixed-mobile convergence，FMC）在20世纪90年代，最初是国际电信联盟为了使IMT-2000移动用户离开归属网络后，能够使用其他网络（如固定网络）继续使用语音业务而提出的，即在没有无线网络覆盖时固网替代接入方式。固定移动融合最初的构想是用户不论在固定还是在移动环境中都能享受到网络提供的相同的服务，获得相同用户体验。ITU和ETSI对固定移动融合的定义基本相同：FMC是在一个给定的网络环境中，运营者不用考虑用户接入技术和位置，通过一种机制，可以为用户提供一系列连续的服务，网络提供服务的能力与用户的固定/移动接入技术以及用户位置无关固定移动融合网络架构。FMC体现了网络向终端用户提供服务时不受接入技术的限制，这包含了网络的融合、业务的融合、产业的融合和终端设备的融合4方面的含义。

（1）网络融合。传统的移动网络与固网的运营是分离的，两个网络的基础设施都是相互独立的，用户获取移动接入与固定接入方式也是完全分开的。固定移动网络融合意味着为固定、移动两个网络构建统一的业务处理、控制的核心架构，能最大限度地利用两个网络原本的通信基础设施，同时提供移动和固定接入服务。网络融合重点体现为核心控制层面的融合与接入层面的无关性，能够提供各种接入方式的网络服务，网络融合的具体表现是应能为用户提供在固定条件下的ADSL、PON接入，移动条件下的2G/3G/4G/WLAN等各种接入方式的无差别的网络服务与业务体验。

（2）业务融合。用户有越来越多的网络业务可以选择，但是提供给移动或固网用户

的业务是独立的。业务融合的好处在于,运营商可以综合运用固定移动技术及基础设施,推出一系列标准统一的业务组合,含有话音、音视频、游戏、可视电话以及 IPTV 等业务,终端用户则可以无须考虑网络的屏障,只是根据需要选择适合的业务。在业务融合的背景下,用户的移动性大大增强,在户内或户外选择最理想的接入方式,一个账单即可享受原来固定、移动两个网络的业务服务。业务融合需要底层数据融合来保障,需要网络融合来支持。

(3) 产业融合。网络与业务的融合必然需要运营商之间企业的融合,这就要求原先分开运作的固网运营商与移动通信运营商之间合作竞争,相互融合。

(4) 终端融合。很显然,为了实现一个终端设备使用多种网络融合提供的不同种类业务,终端需要具备处理多种业务数据的能力,如音视频、游戏、摄影等功能。

1.4 可编程网络

1.4.1 认知无线电与认知网络

1. 软件无线电

经过几十年的发展,无线通信有了长足的进步,出现了多种调制与接入方式不同、功能各异的系统,包括卫星通信系统、蜂窝移动通信系统、无线寻呼系统、短波通信系统、微波通信系统等。然而,不同的无线通信系统相互不兼容、互通性差的问题也越来越明显,给用户和管理者带来极大的不便。为了解决互通性问题,各国进行了积极的探索,其中一种设想是研制多频段、多功能的电台,用一个系列电台代替其他所有电台。这种想法固然可以解决互通性问题,然而,制造这种全能型设备开支庞大,而且通信技术发展迅速导致设备很快就要更新换代。Joseph Mitola III 博士 1992 年 5 月在美国电信会议上首次提出了软件无线电(software radio)的概念[11],随后在其专著《软件无线电体系结构》中提出软件定义无线电(software-defined radio)的概念。软件定义无线电的本质就是构造一个具有开放性、标准化、模块化的通用硬件平台,将各种功能,如无线电发射/接收、信号产生、调制/解调、编解码、加密模式、通信协议等,用软件编程来完成,并使宽带 A/D 和 D/A 转换器尽可能地靠近天线。这种无线电台是可以用软件控制和再定义的电台,选用不同的软件模块就可以实现不同的功能,而且软件可以升级更新,有些人把软件无线电称为"超级计算机"。软件无线电的概念一经提出就得到全世界无线电领域的广泛关注,其基本结构如图 1-11 所示。

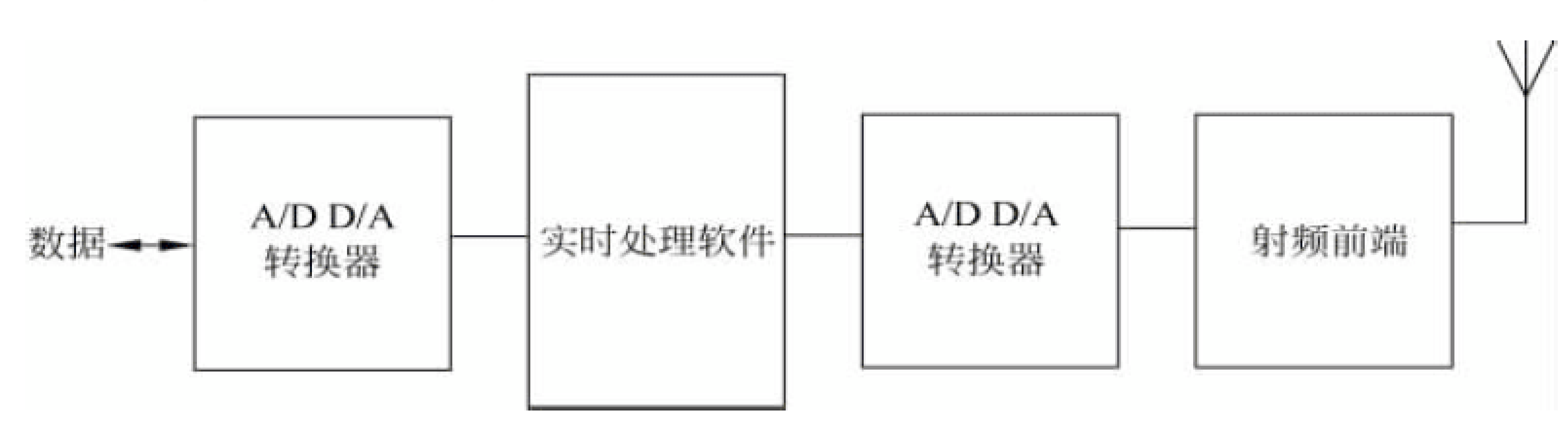


图 1-11 软件无线电基本结构

2. 认知无线电——软件无线电智能化拓展

无线电频谱资源是世界上最稀缺的资源之一，围绕这些稀缺资源的竞争是电信业发展的根本驱动力，提高无线电频谱资源利用率以满足日益增长的无线通信业务需求是无线通信领域永恒的课题。传统的“条块分割”频谱进行静态分配、授权使用的频谱管理策略，由于缺乏灵活性，导致整体频谱利用率不高，这促使了认知无线电的产生。认知无线电通过动态接入未被占用的授权频谱，实现频谱的二次利用，可以缓解频谱资源紧缺的问题。认知无线电是完全可编程的无线设备，它可以感知环境并动态调整其传输波形、信道接入方式、频谱使用和组网协议。如果数字域中的信息是可访问的，则认知无线电背后的驱动力是计算智能算法。认知无线电由软件无线电发展而来，信号处理和机器学习是整个认知无线电的核心，称为认知引擎。究其本质，认知无线电是一种“数学密集型”无线电，它是基于策略的，可以通过认知引擎推出该策略[12]。

人们往往认为认知无线电具有两项基本功能：第一，频谱感知；第二，使用可用感知频谱对无线资源进行“认知”分配。频谱感知是认知无线电的基础，认知无线电将估计或预测频谱可用性和信道容量，并自适应地对自身进行重新配置，以解决在干扰缓解的情况下，实现资源利用率最大。IEEE 1900.6 标准为动态频谱接入和其他高级无线通信系统设计了频谱感知接口与数据结构[13]。分布式感知系统如图 1-12 所示。其中，传感器可以是独立的，也可以是协作传感器的小型网络，通过这些传感器可以推断出可用频段的相关信息；数据档案（data archive，DA）可看作数据库，它存储着频谱使用情况等与频谱感知相关的信息；认知引擎（cognitive engine，CE）是一种包含认知控制机制的逻辑实体，认知控制机制包括推理、方案选择、优化等策略，主要用于自适应无线电控制和频谱

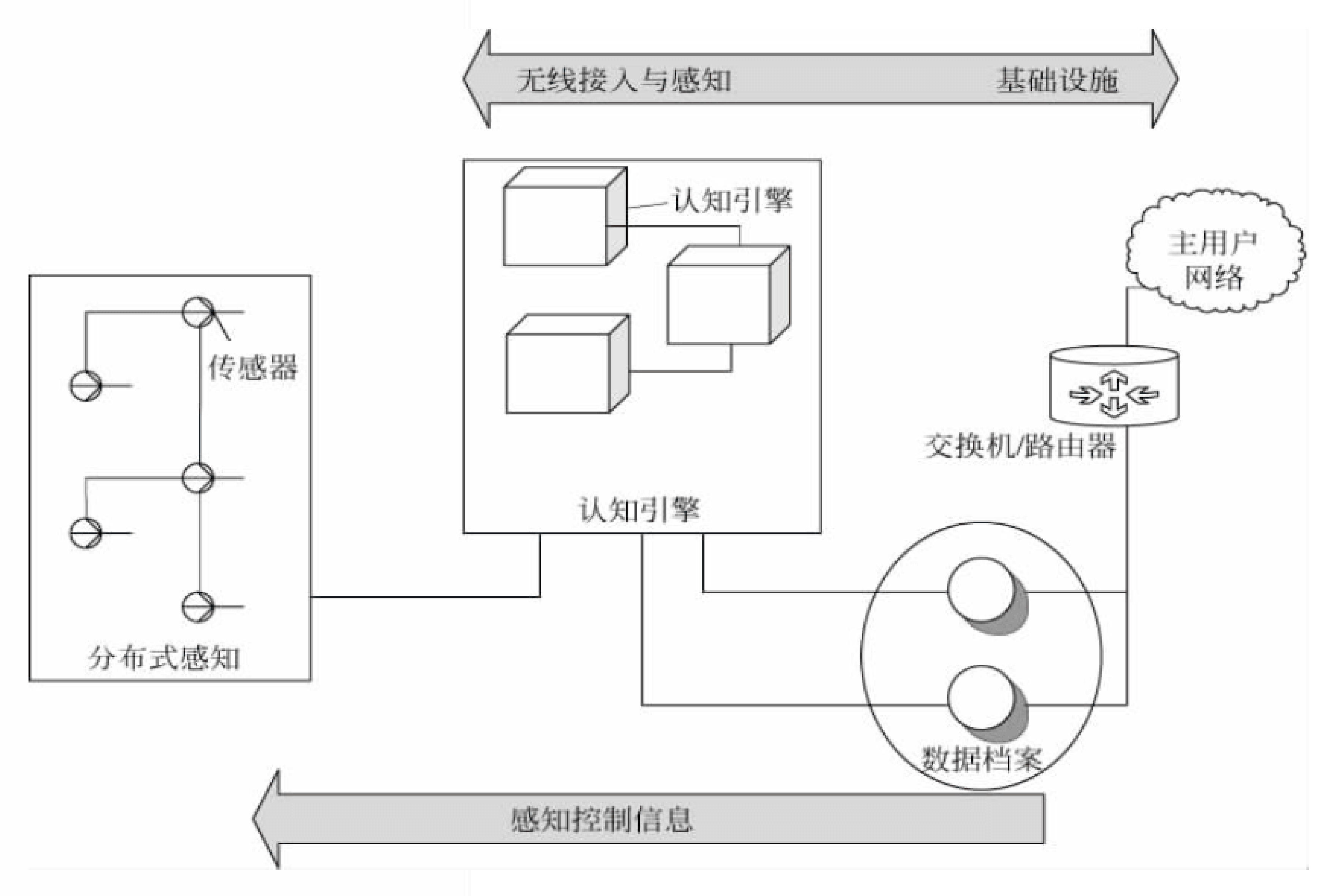


图 1-12　IEEE 1900.6 分布式感知系统

接入。传感器节点在不同的时间、地点、频段以及功能单元分布式感知，在未来的演进方案中，每个连接到互联网的“物体”都可以具有感知功能，例如在物联网和绿色通信中，通过感知获取频谱使用情况的信息并进行实时动态更新，以优化无线网络资源配置。

3．认知无线网络设计挑战

在认知无线电链路形成的网络中，高性能的认知无线电网络必须进行跨层设计，在每一层的设计中都应当了解物理层的频谱情况。由于频谱的动态特性，在设计认知无线电网络上层时，频谱接入新机制会给设计和性能带来诸多挑战。频谱的动态特性对网络各层的影响如下：

（1）MAC 层。由于媒体接入控制层的主要任务是为发射机分配通信资源，而在认知无线网络中通信资源是动态变化的，因此媒体接入控制层为了有效地分配通信资源就必须适应当前的频谱状况，快速地采集与处理信息。

（2）网际层。在传统无线通信网络中，一旦确定合适的路径，这条路径就会被使用很长一段时间。然而，在认知无线网络中，数据路径则需要适应频谱状况。例如，当主用户出现并且阻塞了认知用户的数据路径时，认知用户要么空闲频谱，要么等待原始路径恢复，由于频谱是随机的，因此数据路径也可能是随机的。

（3）传输层。因为次用户利用的空闲频谱具有随机性，造成服务质量不稳定，给传输层带来挑战，因为传输层的部分功能是服务质量控制。

4．IEEE 802.22 标准

2003 年 12 月，美国联邦通信委员会在其规则第 15 章公布了修正案。法律规定“只要具备认知无线电功能，即使是其用途未获许可的无线终端，也能使用需要无线许可的现有无线频带”，为新的无线资源管理技术奠定了法律基础。2004 年 11 月第一个基于认知无线电的无线标准 IEEE 802.22 工作组成立，目的在于解决运营在广播电视频段的感知无线广域接入网络技术。

IEEE 802.22 是第一个基于认知无线电的固定无线区域网络（wireless regional area network，WRAN）标准。工作于 54～862MHz VHF/UHF（扩展频率范围 47～910MHz）频段中的 TV 信道，它可自动检测空闲的频段资源并加以使用，因此可与电视、无线麦克风等已有设备共存。利用 WRAN 设备的这种特征可向低人口密度地区提供类似于城区所得到的宽带服务。802.22 工作频段内，有两类授权用户：一类是电视服务，另一类是无线麦克风。与感知电视信号传输相比，感知无线麦克风要困难许多。通常无线麦克风传输功率在 50mW 左右，覆盖区域在 100m 左右，占用带宽小于 200kHz。由于无线麦克风可能突然地出现和消失，给感知技术带来了巨大的挑战。802.22 系统是固定的一点对多点（PMP）无线空中接口，即基站（BS）管理着整个小区和相关的所有用户终端（CPE）。BS 控制小区内媒体接入和下行（DS）传输给相应的 CPE。而 CPE 通过接入请求，在 BS 允许的上行（US）传输，即任何 CPE 必须在收到 BS 的授权后才能传输。由于 BS 必须兼顾对授权用户的保护，合理地分配感知任务、合理地实现动态频率选择都是 802.22 系统的关键问题。与其他 802 的标准相比，802.22 是针对无线区域网的。BS 的覆盖范围半径最大可达 100km，在目前 4W 的有效全向辐射功率（EIRP）条件下，覆盖面积的半径也

达到 33km。这样 802.22 系统的覆盖面积就是当今最大的一个系统。虽然得益于 800MHz 以下电视频带的良好传播特性，但增加的覆盖范围给技术提供了机遇和挑战。

1.4.2 软件定义网络

目前正在运行的互联网体系架构已经有 40 多年的历史。互联网最初设计的目的是把分散的计算机连接起来以达到资源共享，每个网络设备都有相对独立的操作系统和控制层面，设备与设备之间通过网络协议交换数据。网络配置状态大致总是静态的、不变的，通常是要在一台设备上手动和基于命令行进行配置。随着信息通信技术的发展以及网络规模的不断扩大，互联网一直以来的工作方式导致了一些问题：首先，底层传输技术与应用层技术快速发展，而现有 IP 网络体系架构却没有自动进化能力，通过现有的逐个对网络设备进行配置的方式对网络进行新功能开发效率低下；其次，现有互联网的网络体系结构最初是为固定终端设计的，而随着移动通信技术以及无线接入技术的不断发展，来自移动终端的流量迅速增长，在网络及节点的移动性特征下，端到端的常连接模式不再是互联网通信的最佳方式；最后，现有互联网中的设备，如路由器等网络核心设备，均采用满载设计原则，无论负载量大小如何变化，网络设备都工作在高能耗模式下，这为互联网带来严重的能耗问题。软件定义网络（software defined networking，SDN）的提出正是为了解决传统网络体系结构的问题。

网络中的设备，逻辑上一般是由数据平面和控制平面组成。数据平面通常是一个交换结构，连接了设备上的多个网络端口，主要完成数据处理与转发等工作。控制平面在功能上则相当于大脑。例如，用来在一个网络中构建无环路的路由协议，通常以分布式的方式实现。也就是说，网络中的每个设备都有一个实现了该协议的控制平面。这些控制平面相互沟通协调，以构建网络路径。路由器、交换机、服务器、网关等的协议接口已经存在有多年历史了，几十年来，互联网中对用户、会话和应用程序等相关的动态数据进行编程一直存在并得到了广泛的部署，这些都可以被视为早期形式的 SDN。不同的是，现在大家讨论的 SDN 至少在逻辑上存在一个单独的集中式控制平面，能够在这个平台上编写应用程序，这个超级大脑会将命令推送到每个设备，从而指挥这些设备来操控各自的物理交换和路由硬件，把从不同来源或数据库获得的数据定制化为一个网络范围的操作。回顾互联网的发展过程可以发现软件定义网络的出现是复杂的历史传承，而不是突然出现的技术。

SDN 作为单独的概念首次出现在 2006 年斯坦福大学的 Clean Slate 研究课题，几位工程师创造出了一种称为 OpenFlow 的协议[14-16]。OpenFlow 架构为一些只有数据平面的设备而设计，这些设备被一个（逻辑上的）集中式控制器所控制，这个控制器是该网络中唯一的控制平面。这个控制器负责维护所有的网络路径，以及对其所控制的网络设备进行编程。OpenFlow 协议描述了这些指令和相应规范。2011 年，开放式网络基金会（open networking foundation，ONF）成为第一个致力于 SDN 成长和成功的组织，对 SDN 提供商业上的支持，它的既定任务是把源于学术界的 OpenFlow 协议变为在商业上可行的建设网络和网络产品的基础。两年内，ONF 的成员已发展到约 100 家单位，并且今天仍然是其标准化和市场推广的权威机构。美国、欧洲以及日本等各国一直对 SDN 进行

深入探索与研究，不同的研究机构和标准化组织分别从不同的角度出发，提出了 SDN 参考架构。思科、IBM、微软等设备厂商和软件公司从 SDN 的具体实现和部署的角度出发，共同提出了 OpenDaylight。2012 年，AT&T、英国电信、德国电信等 7 家运营商在 ETSI 发起了一个新的网络功能虚拟化标准工作组 NFV ISG。ITU-T 在 2012 年就开始了 SDN 与电信网络结合的标准研究。

大家对 SDN 抱有不同的兴趣和期望，在此背景之下，很多人都在探寻 SDN 基础理念的更广泛的含义，业内一些人士将 SDN 称为软件驱动网络（software driven network），而不是软件定义网络。在软件驱动的方式中，不是在逻辑上把网络看作集中式控制平面和一些非智能的网络设备，而是将网络看成新旧技术的混合体。软件驱动网络不仅包括 OpenFlow，而且也包括其他形式的网络可编程性，OpenFlow 及其架构被视作所有可能实现 SDN 功能的一个特殊子集。也就是说，完全废除现有的分布式控制的网络，丢弃现有支撑着互联网的先进网络技术，让 ONF 和软件定义网络来构建全新网络是不现实的。更现实的途径是一种混合的方式，网络的一部分被一个逻辑上的集中式控制器操控，而其他部分则由更传统的分布式控制平面来操控。这也意味着，SDN 可以是网络现有工作模式的补充，保留的分布式控制平面的量可以是弹性的[17]，新旧两种工作方式互补协作，实现更广泛、更灵活的网络设备可编程性。

关于什么是 SDN，一直有不同的定义与解读，SDN 技术也在不断发展中。然而人们对于 SDN 最显著的三个特点的认知却是一致的：可编程性、控制平面和数据平面分离以及用于瞬时状态管理的集中式控制模型（即 SDN 控制器）。SDN 控制器所提供的服务能够实现分布式的控制平面所能做到的那些功能，同时它还能实现网络瞬时状态的管理和集中化等概念。最终这些概念体现在一个理想化的 SDN 框架中，而 SDN 控制器就是这个理想化 SDN 框架的具体体现。在现实中，任何一个具体的 SDN 控制器的实例，实际上都是这些功能的一个切片或子集，反映了 SDN 框架对这些概念的取舍。SDN 控制器的具体含义源自其所运行的网络领域，而网络领域的含义则来自于其中所使用的策略和协议选择。

1.4.3 网络功能虚拟化

网络功能虚拟化（network function virtualization，NFV）建立在可编程性、控制和数据平面分离以及 SDN 控制器几个 SDN 主要议题以及虚拟化、数据中心等概念的基础之上。NFV 源于网络运维问题，特别是网络设备厂商在其平台操作系统上捆绑服务带来的影响。网络运维一方面要使计算资源、存储资源以及网络资源构成的现代数据中心的利用率最大化，另一方面在保证不同用户的服务质量的前提下降低网络运行的能耗。

ETIS 给出的 NFV 参考架构[18]如图 1-13 所示，三个主要工作区域包括 NFV 基础设施、虚拟功能网元（virtualised network function，VNF）以及 NFV 管理与编排。NFV 基础设施可分为硬件资源层与虚拟资源层，硬件资源可以来自于不同的网络运营商，可以部署在传统电信机房，也可以部署在数据中心。虚拟功能网元作为网络功能的软件实现，能够运行在 NFV 基础设施上。在 NFV 架构中，一个网络服务可以看作是一系列 VNF 的连接。一个端到端网络服务（例如移动语音/数据、互联网接入、虚拟专用网络）

可以描述为 VNF 与端点的连接图，如图 1-14 所示。从端到端服务的角度来看，一个 VNF 实例对应的实际硬件资源的物理部署情况是不可视的，这使得一个 VNF 实例可以在不同的硬件资源上实现。而对于 NFV 管理与编排来说，出于网络资源管理与配置、网络故障诊断与排除等目的，在任何情况下，VNF 实例和支撑其工作的硬件资源是可视的。

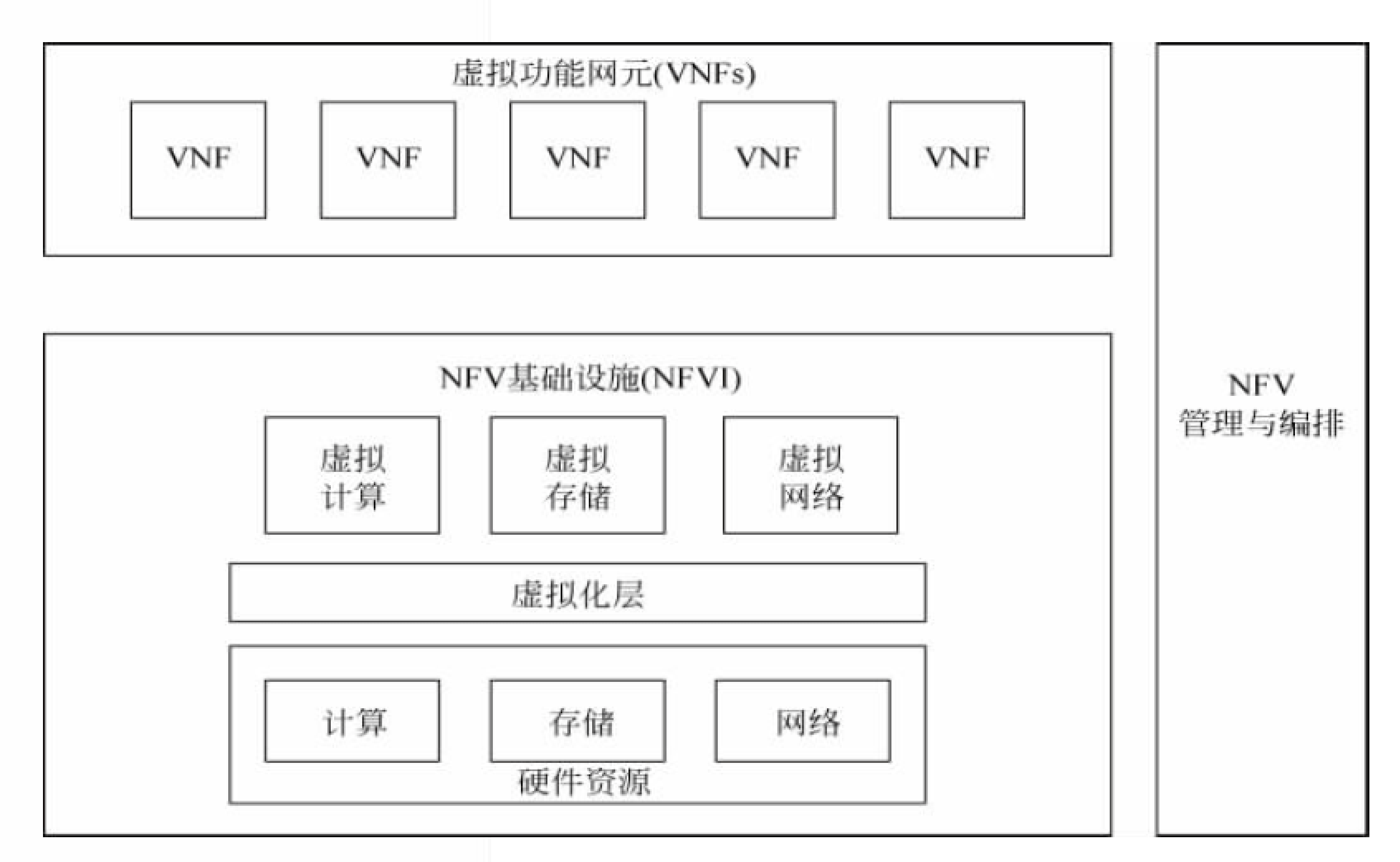


图 1-13　NFV 参考架构

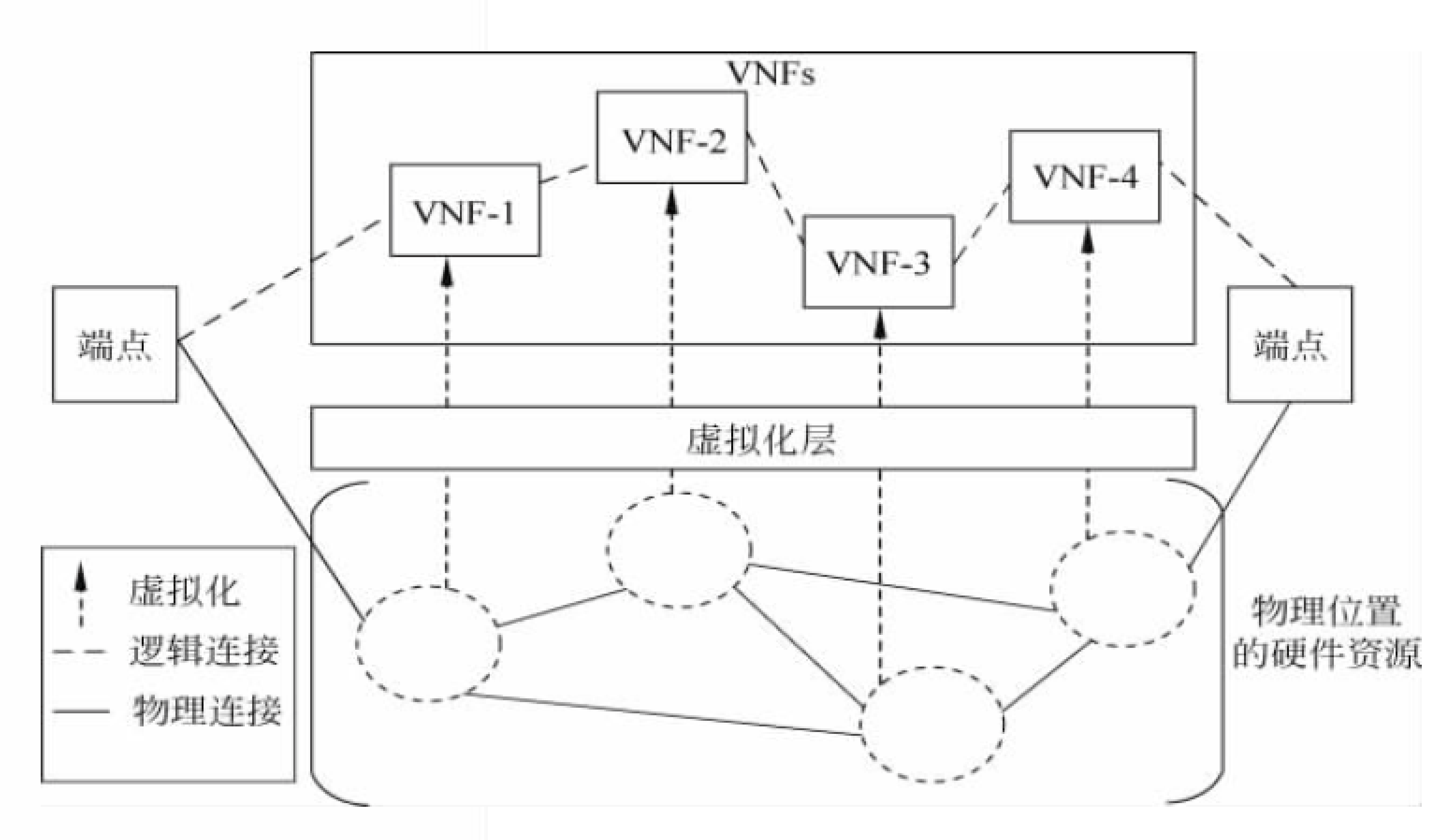


图 1-14　NFV 端到端服务

引入虚拟化层之后，虚拟功能网元与网络中硬件设备完全解耦，改变了电信领域软件、硬件紧绑定的设备提供模式。虚拟资源层对上层应用屏蔽硬件的差异，进而允许运营商对电信系统的硬件资源实行统一管理和调度，这能够有效提升电信网络的灵活性与网络资源的使用效率。网络功能虚拟化之后，电信设备演进为虚拟功能网元，这些网元

的开发和实现将不再依赖于特定的硬件平台，可以降低电信设备（虚拟功能网元）的开发门槛，促进电信设备制造产业链的开放，缩短新业务的部署和推出时间，加速新业务的推出。同时，通过动态地迁移网络负载以及关闭非满载的硬件设备可以有效降低网络能耗。

ETSI是NFV标准化的主要组织，ETSI中的NFV部分以一个工业规范工作组（ISG）的形式存在，由组织主体、技术指导委员会和几个关注特定领域的子组织构成，设备厂商也受邀参与讨论，但从组织设计上来看，官方成员主要是运营商。工作组主要致力于解决关于服务虚拟化的各种问题，目标是定义需求，确定最佳实现，确定当前标准之间的空白，对于如何填补这些空白提出自己的建议。ETSI工作组并非服务虚拟化研究和标准制定的唯一组织。作为SDN研究的一部分，学术界已经围绕NFV的一些议题进行了广泛深入的研究；一些现存的SDN控制器厂商也在分析控制器与虚拟化服务的协作；现有的高度集成网元设备厂商正在尝试把它们的平台以及解决方案完全虚拟化。

1.5 未来网络的特点

1.5.1 智能可编程

当今社会已经是一个网络社会，人们使用的各种设备可以方便地接入网络，人们通过互联网获取最大的便利与收益。网络需要不断地为全体用户提供良好的服务体验，而网络的智能化可编程能够在正确的时间为用户提供正确的服务。网络可编程性已经不是一个新的概念了，从设备组网时代开始网络管理技术就存在。几十年来，互联网中对订阅用户、会话和应用程序等相关的动态数据进行编程确实存在并得到了广泛部署。

传统网络基于端到端通信的概念，即与应用相关的计算和控制工作在发送和接收端进行，网络只完成具有共性的基本处理工作。可编程网络是对传统网络的突破，它赋予网络编程的能力，从本质上将网络中的硬件设施与软件进行分离，在网络节点中提供标准的网络应用编程接口，向用户和网络业务供应者提供一个开放的网络控制机制。可编程网络与传统网络的主要区别在于传统网络是无状态的（stateless），而可编程网络是有状态的并且可以由用户控制和改变。网络应用编程接口允许应用实体利用网络物理资源构建并管理满足自己需求的业务。在网络中通过编程实现对各种业务的控制管理功能，用户可以从更加富有竞争性的市场中获得更有效、更丰富的服务。然而，当要面对一个大系统中的许多台路由器、交换机和服务器时，控制和管理操作需要应用到全网，一个接一个的对网元进行管理就变得不够灵活和敏捷，软件定义网络的出现可以有效解决这一问题。软件定义网络能够在一个平台上编写应用程序，该平台把不同来源的数据定制化为一个网络范围的操作，集中式管理提供整个网络视图，有效提升了网络自动化和编排速度。

网络编程使网络更像一个应用程序，这将孕育出更智能、快捷的应用程序。它使网络以新的方式实现自动化，实时编排网络资源为用户提供更大的灵活性和扩展性，

对于网络业务来说可以有效节约成本并创造收入。智能可编程将会是未来网络的主要特点。

1.5.2 资源最优化

信息通信技术的飞速发展使得网络已经渗透到社会经济生活的方方面面。随着网络规模的不断扩大，网络的异构性、复杂性也越来越明显，而电子商务等大型网络应用的部署和运行则涉及大量静态和动态资源管理。为了保证网络稳定高效的运行，网络管理以及网络资源的合理分配变得异常重要。

互联网中的网络资源既包括路由器、交换机、服务器以及主机等物理资源，也包括网络管理中定义的各种逻辑资源，此外，还包括与网络运行相关的维护人员信息、网络配置信息、网络技术文档以及网络维护人员的经验信息等资源。无线网络中的网络资源包括频率资源、时间资源、码资源、功率资源、存储资源等，相应的网络资源管理内容包括信道配置、功率控制、负载均衡等。

网络资源的管理与利用的目标包含两个方面：从网络运营商的角度出发，提高网络资源的利用率和网络容量；从用户和业务的角度出发，保证用户各种业务的服务质量。为了达到这样的目标，有很多种网络资源管理的方法，具体采用的资源管理技术取决于网络应用场景和业务要求，而最终都是将网络资源进行最优化利用。

网络资源的最优化利用与管理方案几十年来已经得到广泛研究并实施，正在被人们关注的 SDN 以及 NFV 等未来网络技术也在通过动态、灵活配置网络资源，使网络资源得到最优化利用，从而一方面为运营商节约成本，一方面为用户提供丰富高质量的网络服务。资源的最优化利用将是未来网络的一个主要特点。

1.5.3 绿色高能效

在过去的几十年中，无线网络与移动网络发展迅速，越来越多的用户通过移动终端接入互联网，世界范围内已有超过 400 万个基站为用户提供移动接入服务，每个基站年平均耗电量为 25MWh。据统计[19]，2007 年互联网中网络设备、服务器和数据中心总能耗达到 8700 亿千瓦时，占全球总能耗的 5.3%。而随着 ICT 产业的不断发展，网络能耗仍在上升。传统电力能源的过度消耗使得网络运营商不得不支付高额电力费用，同时也产生了大量的碳排放，为生态环境与可持续发展带来压力。为了减轻网络能耗为经济与生态环境带来的压力，绿色高能效已经成为网络技术的主要发展方向。

2008 年 10 月中旬，由气候组织（TCG）代表全球电子可持续性倡议（GESI）完成的《Smart 2020：实现信息时代的低碳经济》报告指出，ICT 行业本身的减排空间巨大，同时，通过信息通信技术的运用，还有助于减少其他行业的碳排放，并为这些行业创造经济效益。已经实施的欧盟第七框架计划关注了通信领域的能源效率与绿色节能问题，主要项目包括 EARTH（energy aware radio and network technologies）、TREND（towards real energy-efficient network design）以及 C2POWER（cognitive radio and cooperative strategies for power saving in multi-standard wireless devices）。在我国，通信业绿色高

能效问题也得到广泛关注，中国通信标准化协会(CCSA)于 2009 年 10 月成立了“通信产品环保标准特设任务组”(ST2)，该特设任务组主要定位在通信领域节能与综合利用的标准体系和框架的研究。

包括 ETSI、3GGP、IEEE 等在内的各大标准化组织已经针对通信网络节能问题开展标准化工作。自 2005 年起，ETSI 的环境工程技术委员会(TC EE)发布了一系列标准，规范信息与通信技术领域的能源消耗。IEEE 也已经开始了高效节能的以太网标准化工作，成立了 IEEE 802.3az 能量有效性工作组推动，该标准使用新的物理层芯片技术，并根据节能需要针对高层协议进行改进。

在各大标准化组织和各国政府的推动下，通信行业各大运营商将节能减排作为降低运营成本的重要手段，设备制造商也针对能量高效技术展开研发。网络设备的能效每隔两到三年就会提高 15%～20%，更新网络设备可以作为实现网络绿色高能效的一种途径。而 SDN 以及虚拟化等新兴技术可以充分考虑网络负载、业务类型以及网络设备的能效特性，动态、灵活地配置网络资源，通过提高网络设备利用率的方式实现网络绿色高能效。另外未来网络还有智能化的特点，包括终端接入的智能化、利用自组织 SDN 等技术的网络拓扑的智能化、结合应用层，通过机械学习提供面向服务的智能化。

参考文献

[1] David Tse and Pramod Viswanath. Fundamentals of wireless communication [M]. Cambridge, England: The Syndicate of the Press of University of Cambridge, 2005.

[2] 吴伟陵，牛凯. 移动通信原理[M]. 2 版. 北京：电子工业出版社，2009.

[3] Cisco. Visual Networking Index, Feb. 2014, white paper [L]. Cisco.com.

[4] A. Sahin, I. Guvenc, and H. Arslan. A survey on multicarrier communications: Prototype filters, lattice structures, implementation aspects [J]. IEEE Commun. Surveys Tuts., 2014, 16(3): 1312-1338.

[5] V. Cerf, R. Kahn. A Protocol for Packet Network Interconnection [J]. IEEE Transactions on Communications Technology, 1974, 22(5): 627-641.

[6] International Organization for Standardization [L]. http://www.iso.org/.

[7] 方德葵. 有线电视网络与传输技术[M]. 北京：中国广播电视出版社，2005.

[8] 顾畹仪. 光纤通信[M]. 第 2 版. 北京：人民邮电出版社，2011.

[9] 刘韵洁. 三网融合与未来网络的发展[J]. 重庆邮电大学学报(自然科学版)，院士专家特别报告，2010, 22(6): 693-697.

[10] Hanrahan. Network Convergence: Services, Application, Transport and Operation Support [M]. John Wiley and Sons Ltd, Jan. 2007.

[11] J. Mitola. Software radios: Survey, critical evaluation and future directions [J]. IEEE Aerospace and Electronic Systems Magazine, 1993, 8(4): 25-36.

[12] Robert C. Qiu, Zhen Hu, Husheng Li, Micheal C. Wicks. Cognitive radio communications and networking-principles and practice [M]. John Wiley and Sons Ltd. 2012.

[13] IEEE Std 1900.6-2011. IEEE Standard for Spectrum Sensing Interfaces and Data Structures for Dynamic Spectrum Access and other Advanced Radio Communication Systems. 2011: 1-168.

[14] Stanford University. Clean slate program [L]. http://cleanslate.stanford.edu/

[15] McKeown N., Software-Defined networking [L]. In: Proc. of the INFOCOM Key Note. http://

infocom2009. ieee-infocom. org/technicalProgram. htm

[16] McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J. Open Flow: Enabling innovation in campus networks [J]. ACM SIGCOMM CCR. 2008, 38 (2): 69-74.

[17] Thomas D. Nadeau, Ken Gray. SDN: Software Defined Networks [M]. O'Reilly Media, 2013.

[18] ETSI. Architectural Framework [L]. http://www. etsi. org/deliver/etsi_gs/NFV/001_099/002/01. 01. 01_60/gs_NFV002v010101p. pdf.

[19] C. Hu, C. Wu, W. Xiong, B. Wang, J. Wu, and M. Jiang. On the design of green reconfigurable router toward energy efficient internet [J]. IEEE Commun. Mag. 2011, 49(6): 83-87.

第2章 异构无线网络融合方案

本章对异构无线网络融合的体系结构等进行研究，首先介绍业务层融合的两个协议：IMS和SIP，接下来介绍IP层和物理层融合的热点技术，最后对组网融合和移动终端融合的关键技术进行简要说明。

2.1 异构无线网络融合的体系结构

本节首先介绍3GPP系统与WLAN网络的两种融合方案：紧耦合方案和松耦合方案，并说明了6种融合场景，接下来对描述了无线接入网融合的新型架构C-RAN及其演进路线。

2.1.1 3GPP系统与WLAN网络融合体系结构

近年来，以802.11标准为主的WLAN以其低廉的建网价格以及高传输带宽(802.11系列标准能提供1～54Mb/s的数据传输速率)迅速拓展市场空间。但是它的缺点也很明显，每个接入点的覆盖范围不大，只能适用于公司、旅馆、机场等所谓的“热点”地区，而且不同WLAN业务提供商之间的网络没有漫游协议。而3GPP则能弥补WLAN这些所有的缺点：3GPP作为一个完整的移动通信系统可以为用户提供无所不在的连接性，在不同的PLMN之间有成熟的漫游协议。但是3GPP的投资规模庞大，数据峰值传输速率也只有2Mb/s左右。

由于WLAN和3GPP这些互补的特性，3GPP开始研究3GPP-WLAN的互通，并确定互联的基本原则：3GPP-WLAN互通必须尽量减少对WLAN以及3GPP标准的影响；既保持WLAN标准不变，同时使对3GPP现存规范的修改最小化。

针对WLAN同3G网络融合的研究和标准化工作已在世界范围内积极开展，其中具有代表性的主要是WLAN和3G各自的标准化组织。欧洲的ETSI最初针对HiperLAN2提出了松耦合和紧耦合两种方案。负责GSM和UMTS维护及标准发布的3GPP也于2003年开

始互联标准化工作，从其协议版本 R6 开始逐步提出网络融合的解决方案。考虑到 3G 网络的应用将更为广泛，以下主要从 3GPP 网络(GPRS/UMTS)的角度讨论其签约用户如何利用 WLAN 资源以更低的成本开展更加高速的数据业务。下面分别从融合方案和融合场景两个方面进行讨论。

1. 3GPP 系统与 WLAN 网络融合方案

1）紧耦合

“紧耦合”的原理是将 WLAN 网络作为 3G 网络的接入网，并成为 3G 核心网的一个组成部分[1]，因此这里的 WLAN 网络自然要具有 3G 接入网络的功能。若该 3G 网络为 CDMA2000 核心网，那么“802.11 网关”以 PCF 形式上溯到 3G 核心网中。而在 WCDMA 核心网中，“802.11 网关”则作为该网络的 SGSN 部分。同时，“802.11 网关”隐藏了作为独立的 WLAN 网络的细节，实现了 3G 接入网要求的所有协议规范。在该方案下，移动点要在它们的标准 802.11 网卡顶层实现 3G 协议栈，以便在不同网络的物理层路由。WLAN 网络中由客户机程序产生的业务流量都要注入 3G 核心网的协议。不同网络共享相同的鉴权、信号发射、传输，以及报表等机制结构，并独立于应用在无线接口处的物理层协议。

然而，上述方案却有一些不足之处。既然 WLAN 网络接口要直接与 3G 核心网互连，那么同一运营商必须同时拥有 WLAN 和 3G 两个网络部分。事实上，独立控制的 WLAN 不能与 3G 网络融合。如今 3G 网络开发利用的是周密的工程化的网络计划工具，每个网络元素的容量和配置规划都要比空中接口上的技术实现更加规范。为了将 WLAN 网络中的数据流量注入 3G 核心网中，整个网络中的各部分如 PDSN 和 GGSN 的配置设计都要修改以适应负荷的不断提高。此外，用户使用的接入设备的配置也会出现一些问题。要在 802.11 网卡顶层实现 3G 协议栈，需要具备基于用户服务识别模块(USIM)和用户可分离识别模块(RUIM)的具体 3G 鉴权机制。为了在 WLAN 网络中进行鉴权操作，要求 WLAN 网络提供商将网络接入到 3G 运营商的 7 号信令网中进行鉴权过程。因此，用户设备的 WLAN 网络接口要植入到 USIM 模块或 RUIM 模块中。

综上所述，“紧耦合”这种方案要求重新配置 3G 核心网络和 802.11 网关的复杂性以及随之而来的高额成本，迫使网络设计者选择“松耦合”互连模式实现 WLAN 与 3G 之间的融合。紧耦合的结构如图 2-1 所示[2]。

紧耦合方案有以下几个优点：

(1) 重用 3GPP 的用户信息，核心网资源以及认证、计费和授权系统，无须对现有 3GPP 网络进行改造，保护了 3GPP 运营商的投资。

(2) 基于 3GPP 的会话和移动性管理，使用户在 WLAN 和 3GPP 网络间的切换时延小，丢包率低。

(3) 支持基于 3GPP 分组域的业务(短消息、定位等)访问。

(4) 可以依照具体策略在 3GPP 小区和 WLAN 之间进行负载平衡。

(5) 在 WLAN 对底层数据加密的基础上，使用 3GPP 的认证和加密机制，增加了 WLAN 的安全性。

但紧耦合方案也有如下几个缺点：

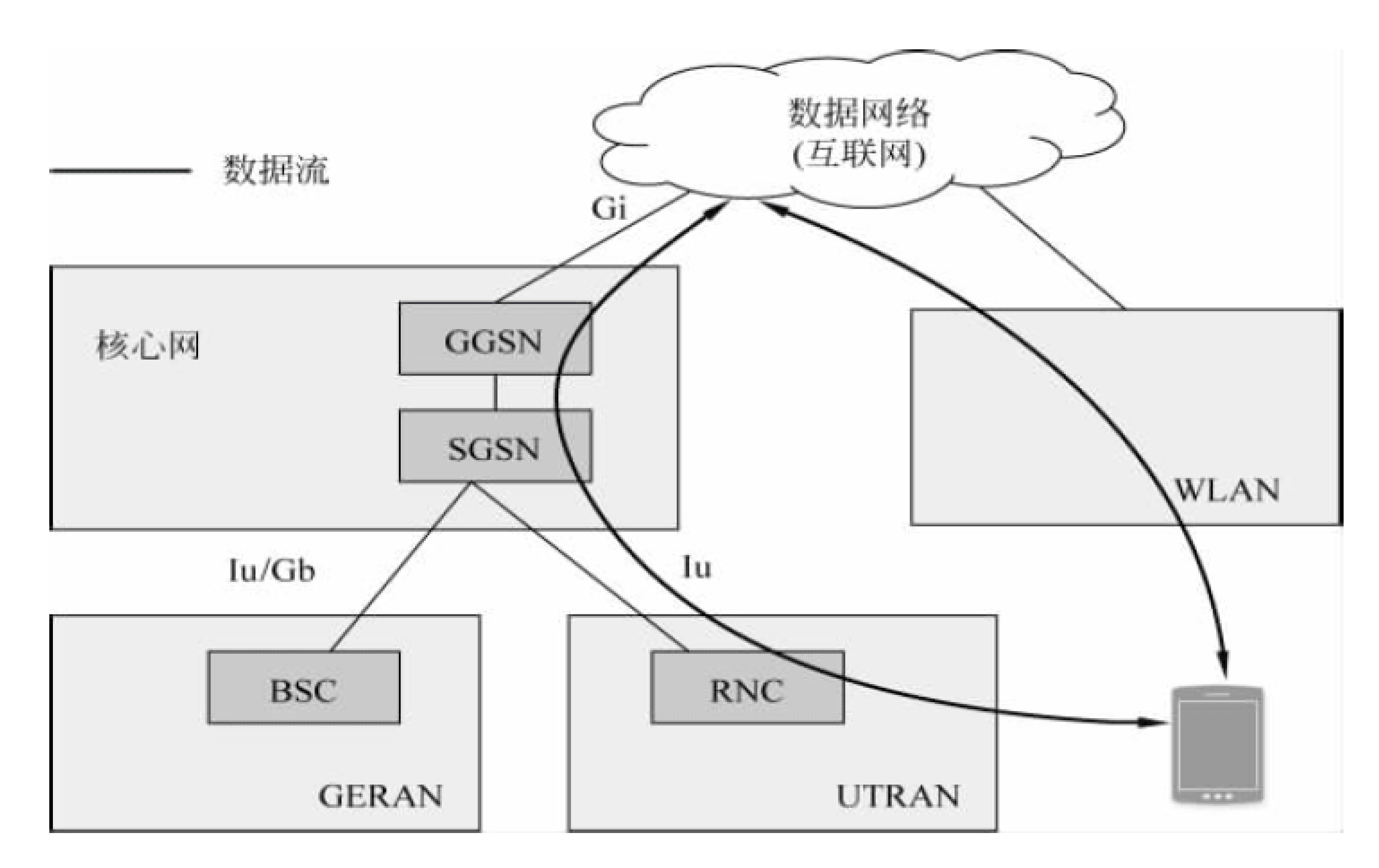


图 2-1　紧耦合的结构示意图

(1) 用户设备以及 WLAN 网关需要实现必要的 3GPP 协议栈,复杂度高。

(2) 由于 WLAN 仅作为 3GPP 信令和数据的承载方式,传统的 WLAN 设备无法通过 WLAN 直接访问网络业务。

(3) WLAN 和 3GPP 结合得过于紧密,所有的信令和数据都由 3GPP 网络承载,加重了网络负担,容易形成网络瓶颈。

(4) 通常要求 WLAN 和 3GPP 网络由同一运营商经营,不适宜大范围应用。

2) 松耦合

如同前面的结构,"松耦合"模式方案在网络融合中也引入了 802.11 网关。但是,在本方案中,802.11 网关是和互联网直接相连,而没有直接连接到 3G 网络中的 PDSN、GGSN 或 3G 核心网路由器上。接入 802.11 网关的用户可能来自本地网络或者其他网络的访问者。之所以称这种方案为"松耦合"互连模式,是因为它将 WLAN 网络和 3G 网络的数据传输路径完全分离。高速的 WLAN 数据流不会进入 3G 核心网,但终端用户仍可以实现网络的无缝接入。其结构如图 2-2 所示。

在这种方案下,不同机制和协议都能处理网络中 3G 和 WLAN 节点的鉴权、结算和移动管理,并且为了无缝操作还要相互合作。在具体操作中,WLAN 与 CDMA2000 融合时,要求 802.11 网关能支持移动 IP 功能以处理网络间的移动性问题以及与 3G 本地网的 AAA 服务(鉴权、授权、付费)的合作。这样,3G 运营商就能够收集 WLAN 的结算记录,从而生成一份统一的报表清单。同时,在这两个网络中采用一致 AAA 服务允许 802.11 网关动态地从本地 AAA 服务中获得每个用户的服务策略,并使该服务策略适用于 WLAN。而在 WLAN 与 WCDMA 融合时,由于 WCDMA 标准还不包括支持诸如 AAA 和移动 IP 的 IEIF 协议,所以 WLAN 与 WCDMA 融合时要求有更多的适配性。移动 IP 服务要先做更新再接入 GGSN 以便在融合后的网络中无缝漫游。为了在网络中 WCDMA 部分鉴权和生成报表,通用用户数据库要连入本地寄存器。此外,当客户在 WLAN 中漫游时 AAA 服务数据库也同样要与本地寄存器互连。

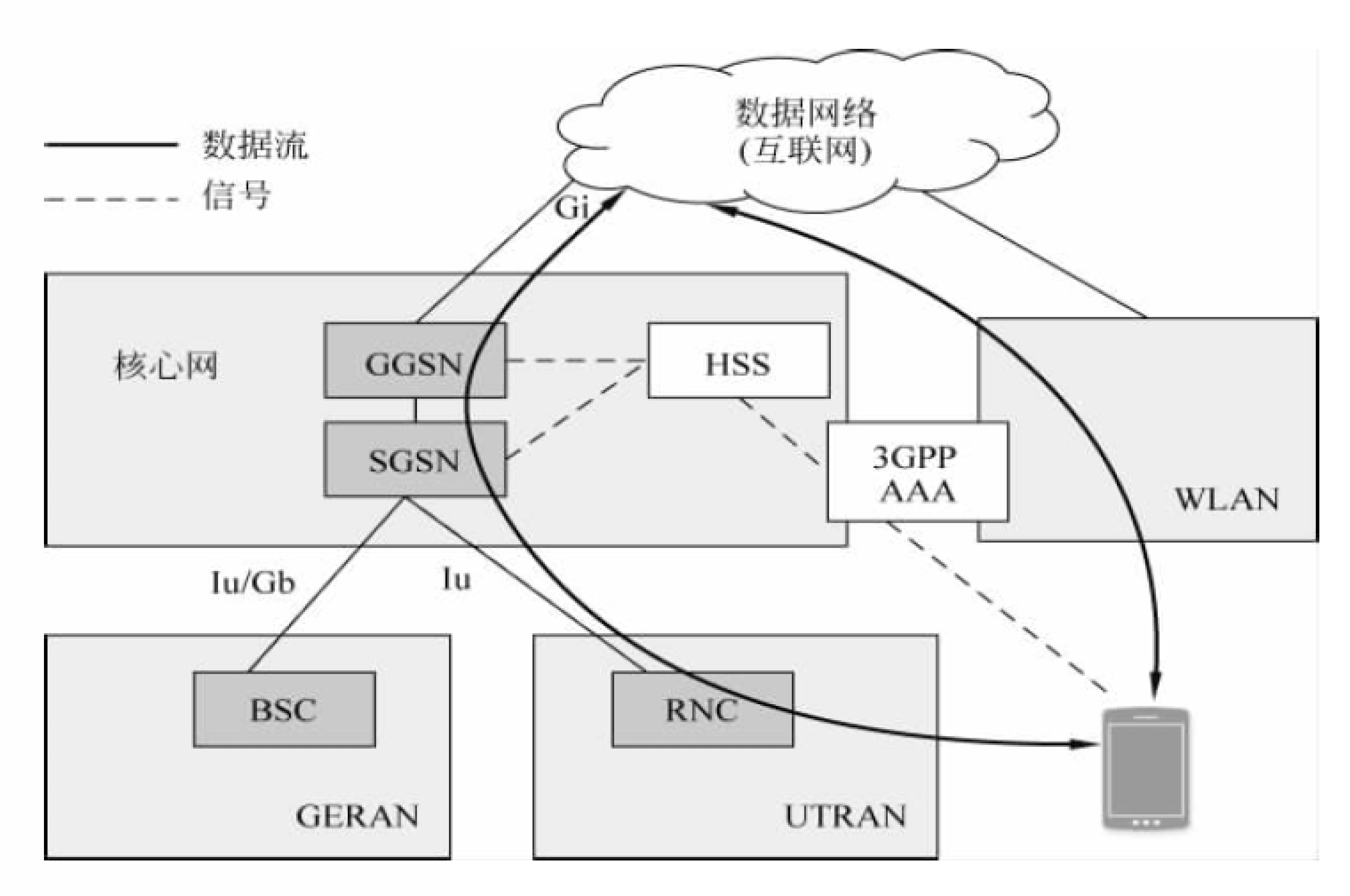


图 2-2　松耦合的结构示意图

从上面的论述，可以看到"松耦合"互连模式方案具有一定的优势。首先，它允许WLAN 和 3G 独立发展，且不需要重新修改各自的核心结构，3G 运营商不用投入额外的资金就能从其他 WLAN 服务供应商中获益，同时利用完善的工程技术手段继续发展已有的网络；其次，与众多合作伙伴达成漫游协议的结果会使网络覆盖范围更加扩大，用户可以在热点地区通过一个网络运营商获得所有网络的接入服务，运营商也不必为不同地区、不同接入网络建立独立的业务结算服务，从而大大提高网络的利用效率；最后，与紧耦合模式不同的是，松耦合模式结构允许 WISP 通过漫游协定使得其提供的 WLAN 与其他公共 WLAN 和 3G 网络自由互连。

但与紧耦合方案相比松耦合方案还有以下几个缺点：用户设备需要实现移动 IP 以支持网络间的切换；移动 IP 自身的缺陷会引起较大的切换时延，不利于实时业务的切换；WLAN 环境下不支持基于 3GPP 分组域的业务访问(除 Internet 业务)。

2. 3GPP 系统与 WLAN 网络融合应用场景

为了使 3GPP-WLAN 网络融合的实现更加灵活、通用、可扩展，3GPP 于 2003 年开始"3GPP 网络同 WLAN 互联"的标准化工作，在 3GPPTR22.934 中定义了 3GPP 网络和 WLAN(在这里，把 WiMAX 网络当作一种 WLAN)互通的 6 种场景，这些场景很好地描述了融合的层次[3]。

情景 1：公用计费和客户关系

即统一计费和客户关系模式。用户签约两个网络业务，并能随意接入两个网络，但各种费用采用一个清单结算。情景 1 要求的规格没有影响 3GPP。

情景 2：基于 3GPP 系统的接入控制和计费

即基于 3GPP 网络接入控制和计费，由 3GPP 系统提供统一的认证、鉴权和计费的功能。WiMAX 网络与 3GPP 网络的用户标识、鉴权方式、授权内容并不相同，但可以通过维持两网用户标识映射关系，使用相同的 AKA 鉴权方式，针对两套系统提供有差异的授

权信息来实现两套系统在接入控制层面的融合。

情景3：接入3GPP的分组交换域业务

共享3GPP网络PS业务，用户通过WiMAX的接入可以访问3GPP网络的PS业务，如LCS、MMS、即时消息等。其实这一阶段的融合意义并不是很大。首先，WiMAX和3GPP的PS域最基础的业务是分组数据接入业务，最基础的业务已经为两个系统默认提供了。其次，WiMAX Forum在致力于制定WiMAX的支撑业务，如LBS、MBMS、PCC等，WiMAX可以独立演进一套更适合自身的支撑业务。

情景4：业务连续

用户可以在两个网络中进行业务的切换，业务能够保持，允许停顿和丢包。如HTTP、即时消息业务、呈现业务允许一定的停顿和数据包的丢失。

情景5：无缝业务

用户可以在两个网络中进行业务的切换，切换过程中用户感受不到业务的停顿和数据包的丢失。实时业务对业务连续性的要求很高，如VoIP、流媒体、IPTV等。而在现阶段，在实际应用中，以WCDMA为代表的3G技术的带宽还不足以普遍开展这类业务，因此，这一层次的融合需求并不迫切。

情景6：接入3GPP电路交换域业务

共享3GPP网络CS业务，通过WiMAX的接入，用户可以直接使用3GPP网络的CS业务，并提供无缝的业务切换能力。这一阶段的融合不适合在接入网实现，复杂度太高，适合在更高的网络层次来实现，如IMS、soft switch。

2.1.2 无线接入网络融合[4]

1. 绿色无线接入网技术架构

C-RAN是根据现网存在的问题和技术的进步，提出的新型无线接入网架构。其中的C有4重含义，即集中化处理(centralized processing)、协作式无线电(collaborative radio)和实时云计算架构(real-time cloud infrastructure)的绿色无线接入网架构(clean system)。集中式基带处理可以大大减少覆盖相同区域的基站数量；面向协作的无线远端模块和天线可以提高系统频谱效率；基于开放平台的实时云型基础设施和基站虚拟化技术可以降低成本，共享处理资源，减少能源消耗，提高基础设施利用率。C-RAN的总体目标是为了解决互联网的飞速发展给运营商带来的多方面压力(能耗、建设和运维成本等)，建设绿色可持续发展的网络。

C-RAN架构主要包括三部分：由高性能通用处理器和实施虚拟技术组成的集中式基带池、由远端射频单元(RRU)和天线组成的分布式无线网络、连接RRU和集中式基带池的高宽带低延迟的光传输网络[5,6]，如图2-3所示。采用C-RAN架构可以实现基站设备的虚拟化，从而实现资源的高效利用和能源的低消耗。

C-RAN有以下几点优势：

(1) 节约OPEX和CAPEX成本。C-RAN是一个绿色网络。通过集中式基站处理单元，可以使机房数量至少减少一个数量级，有效地降低了由于房屋租赁和基站建设造

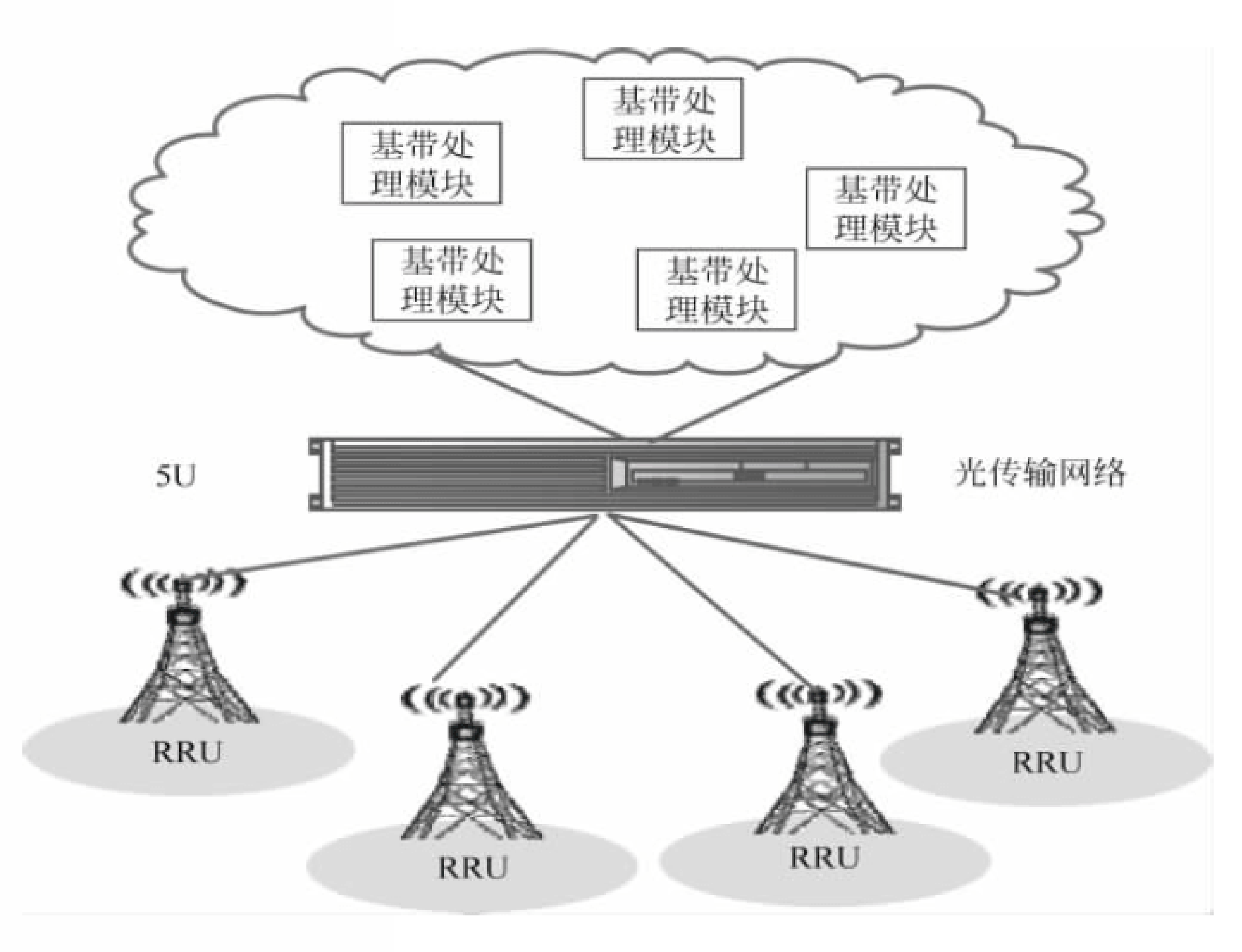


图 2-3　C-RAN 架构图

成的成本，同时降低了运营及维护成本。

(2) 降低能耗。首先，集中化的处理方式可以大大减少机房数量，从而大大地减少空调的能耗；其次，高密度的射频单元可以缩短其与用户终端之间的距离，在覆盖范围不变的情况下，减少网络和用户的发射功率；最后，因为所有基站共用一个资源池，通过资源的动态调度可以处理不同的 RRH 的基带信号，从而降低了潮汐效应带来的能耗。

(3) 提高网络容量。在 C-RAN 中，虚拟基站可以共享所有通信用户的很多信息，如接收和发送信息、业务数据和通道质量等。这样更利于实现资源的联合处理和调度，从而提高频谱的使用效率。

(4) 基于负载的自适应资源匹配。由于 C-RAN 中资源的联合调度，使得网络可以随着各个时段和位置的不均衡负载灵活地调配处理资源。用户在小区间进行移动时，其占用的基站资源也是随之移动的，故不会对资源利用率造成影响。

(5) 互联网业务的智能减负。C-RAN 可以采用智能减负技术，使得用户产生的大量互联网业务被移出核心网，从而使传输网和核心网的业务负载和相应成本降低，并给用户带来更好的服务体验。

2. 演进路线

新型的 C-RAN 是传统无线接入网络的一次变革，但传统的无线接入网络不可能一夜之间被改变，故 C-RAN 以逐步演进的方法逐步取代传统无线接入网络，其演进过程分为三个部分。

1) 基于光传输网络的分布式 BBU+RRU 基站

基站的功能可以由分离的远端无线射频单元(RRU)和基带单元(BBU)来实现，并且

可以实现集中化基站内的多个 BBU 的基带处理资源在载波级上的调度。RRU 通过光纤或传送网与 BBU 相连接，可以部署在远离 BBU 物理位置的远端站点。为方便部署，RRU 设计的小巧而轻便，通过光纤从 BBU 接收，或给 BBU 发送无线电信号，其中 BBU 是信号处理的核心。类似 OBRI、RRU 和 BBU 之间的光纤连接也可以被标准化，使得不同厂家的 RRU 和 BBU 可以互联。

集中式基站内具有一个高容量、低延迟的交换矩阵以及相关协议支持多个 BBU 内的载波处理单元之间的互联互通。分布式 RRU 的载波基带信号可以交换到集中式基站中的任意一个 BBU。由此，集中式基站可以有效地实现载波负载均衡，避免了 BBU 过度负载或者过度空闲的现象。由此可以实现更大范围的载波负载均衡，提高设备利用率，降低能耗，并可以更方便地部署协作式 MIMO 以及干扰消除等信号处理算法，从而增加无线系统的性能增益。

2）基于通用处理器和协作式处理的基站

BBU 的基带处理完全由基于通用处理器（GPP）的软件无线电（SDR）实现。这与传统基站中由 ASIC、FPGA 或 DSP 完成基带处理的实施方案不同。虽然 FPGA/DSP 可提供一定程度的灵活性，但它们不是通用处理器，因此难以保证后向兼容性，并且，FPGA/DSP 的 HW/SW 设计通常只能专用于特定的硬件结构。将基带处理由通用处理器上的软件无线电实现，可以更容易地支持多标准或升级 HW/SW，方便支持新标准的引入、增加处理容量。

通过将多个 RRU 连接到集中式的 BBU，并在 BBU 中实现软件无线电，协作式波束成型和联合信号处理将更容易在 BBU 中实现。多个基站通过交换调度信息、信道信息和用户数据，可使得系统容量最大化，并减小了系统干扰。

3）基于实时云架构的虚拟基站

一旦 BBU 建立在通用处理器上，并可以用软件无线电实现基带处理，那么基于实时云架构的虚拟基站将是 C-RAN 演进的下一个方向。

大量通用处理器通过高带宽、低延迟的互联网架构，可组成一个巨大的计算池——“实时云计算基带池”，类似于 IT 产业中的广泛应用的“云计算”系统。其不同点在于：实时云计算基带池所实现的虚拟基站的基带处理任务是实时进行的。可以配置的 RRU 通过收发无线信号，来实现多种标准的无线网络的覆盖。光纤和负载转换器连接 RRU 和虚拟基站，将 RRU 传送的基带信号发送到指定的虚拟基站。虚拟基站则利用实时云计算基带池动态分配的计算资源以完成无线基带信号的实时处理。同时，在具有多个实时云计算基带池的系统中，为了平衡不同实时云计算基带池之间的计算负载、满足处理时延的要求，负载均衡器应该具备转发 RRU 基带信号到指定实时云计算基带池的能力。

2.2 异构无线网络的业务融合

2.2.1 基于 IMS 的全业务网络融合

IP 多媒体子系统（IMS）由第三代移动通信合作计划（3GPP）组织在 R5 版本中提出，是对 IP 多媒体业务进行控制的网络核心层逻辑功能实体的总称。现在通常提到的 IMS

实际有狭义和广义两种概念，狭义特指 3GPP R5 版本以后的网络核心控制层所涉及的逻辑功能实体，广义是对基于 IMS 网络架构的统称。

1. IMS 国际标准研究

对 IMS 进行标准化的国际标准组织主要有 3GPP、电信和互联网融合业务及高级网络协议（TISPAN）。3GPP 侧重于从移动的角度对 IMS 进行研究，而 TISPAN 则侧重于从固定的角度对 IMS 提出需求，并统一由 3GPP 来完善，最终实现 IMS 对固定接入和移动接入的统一控制。

3GPP 采用和其他标准化组织密切合作的方式制定 IMS 技术标准，并依靠各国通信界的协同不断发展和完善。其中，最重要的合作就是和 IETF 联合开发了 IMS 网络的核心控制协议 SIP，共同研究 SIP 的扩展和应用。

3GPP 依照 UMTS 3G 移动通信系统的系列标准版本定义了 IMS 的分阶段演进计划[5]。3GPP 在 R5 版本中首次提出 IMS，并在 R6 和 R7 版本中进一步完善。IMS R5 版本在 2002 年 9 月已经冻结，它侧重于对 IMS 基本架构、相关功能实体、功能实体之间的交互流程等进行研究；R6 版本在 2005 年 3 月冻结，R6 版本阶段则更侧重于 IMS 和外部网络之间的互通（包括和电路域网络的互通、和 IP 网络的互通、和无线局域网的互通），以及 IMS 支持各种业务的能力，并明确业务由 IMS 用户的归属地提供和控制，使 IMS 真正成为一个可运营的网络技术；R7 融合 TISPAN 关于 NGN 的研究标准，增加对于数字用户线路接入技术的支持，提出策略控制与计费（PCC）技术以及 WLAN/3G 语音连续通信的话音连接连续（VCC）技术。正在研究中的 R8 将协同各个标准化组织的工作，研究支持组合业务的业务代理技术以及下一步的统一 IMS 架构。

2. IMS 分层网络结构

IMS 分层网络结构分为底层、中间层、上层。

底层为以 IPv4/IPv6 为核心技术的承载层，包括话音、视频、数据等各种媒体在内的数据流，既可经由 2G/3G 移动网接入技术接入网络，也可经由 WiFi、DSL 等固定网宽带接入技术接入网络，但所有媒体流均以 IP 分组的形式在网络中传输，网络运营商构建统一的 IP 宽带网络为所有用户服务，体现了以 IP 为核心的网络融合。

上层为以信息技术为核心的业务/应用层，业务运营商通过部署在各类应用服务器中的业务逻辑向端用户提供增值业务，业务提供独立于用户的接入技术，体现了以信息与通信技术（ICT）为核心的业务融合，并通过 API 调用融合通信网的能力。

中间层就是以 SIP 为核心技术的 IMS 控制层，向下对传送层的 IP 多媒体通信进行呼叫控制和管理，向上为业务应用层提供调用通信网能力的开放式接口，实现业务层和网络层的分离，支持独立于网络运营商的业务运营商的形成，体现了多媒体业务的融合控制与管理。

3. IMS 实现网络融合的优势

利用 IMS 实现对固定接入和移动接入的统一核心控制，主要是 IMS 具有以下特点：

（1）与接入的无关性。虽然 3GPP IMS 是为移动网络设计的，TISPAN 是为固定

xDSL 宽带接入设计的，但它们采用的 IMS 网络技术却可以做到与接入无关，因而能确保对 FMC 的支持。从理论上可以实现不论用户使用什么设备、在何地接入 IMS 网络，都可以使用归属地的业务。控制层和业务层之间的开放接口使得来自不同接入网络的呼叫/会话可以进行统一的处理。

（2）统一的业务触发机制。IMS 核心控制部分不实现具体业务，所有的业务包括传统概念上的补充业务都由业务应用平台来实现，IMS 核心控制只根据初始过滤规则进行业务触发，这样消除了核心控制相关功能实体和业务之间的绑定关系，无论固定接入还是移动接入都可以使用 IMS 中定义的业务触发机制实现统一触发。

（3）开放的业务环境。IMS 的业务除由运营商自己提供以外，还允许由第三方提供。IMS 提供标准接口，业务开发基于 API 而不是直接面对复杂的网络协议，屏蔽了网络协议的复杂性。

（4）一致的业务归属能力。在 CS 域和 PS 域以及 PSTN，业务提供能力都与用户当前所在的设备有关系。在归属网络，已经开通的业务在漫游地设备并不一定能够提供。IMS 在这方面有很大的变化，所有业务处理信令都要回到归属网络，业务环境也是由归属网络提供的。

（5）统一的用户数据库。HSS（归属业务服务器）是一个统一的用户数据库系统，既可以存储移动 IMS 用户的数据，也可以存储固定 IMS 用户的数据，数据库本身不再区分固定用户和移动用户。特别是业务触发机制中使用的初始过滤规则，对 IMS 中所定义的数据库来讲完全是透明数据的概念，屏蔽了固定和移动用户在业务属性上的差异。

IMS 所具有的这些特征可以同时为移动用户和固定用户所共用，这就为同时支持固定和移动接入提供了技术基础，使得网络融合成为可能。

4. 以 IMS 为核心的 FMC 技术

1）网络融合技术

IMS 网络融合的基本思想是将互联网 IP 技术、源于固定网演进的软交换技术和移动核心网技术有机地结合起来，实现固定和移动通信网融合的目标。

首先，IMS 鉴于 VoIP 的成功应用，在承载层摒弃了长期以来在传统电信网中占主导地位的电路交换（CS）域，采用基于 IP 技术的单一的 PS 域传输各种媒体和信令数据流，即首次提出在通信网中采用全 IP 承载网络，使之成为通信网融合的底层基础。

其次，在业务组网上继承蜂窝移动通信系统特有的网络技术，沿用原籍网络和访问网络的概念，继续采用扩展的移动性管理技术以及集中设置的网络数据库支持用户漫游和切换。在 2G 网络归属位置寄存器（HLR）的基础上扩展形成 IMS 至关重要的网元原籍用户服务器（HSS），存储包括位置信息、认证授权信息、用户文档以及归属 SIP 服务器在内的处理多媒体会话所需的各种用户信息。因此，IMS 系统在灵活提供 IP 多媒体业务的同时仍保持移动通信的所有特点，体现了移动通信网络技术和 Internet 技术的有机结合。

另外，在异构网络互通和异构终端接入上借鉴软交换组网技术，通过网关实现与传统固定及移动网络的互通，设有信令网关（SGW）、媒体网关（MGW）、媒体网关控制器

(MGCF)等网元，而且在 MGCF 和 MGW 中也采用 IETF 和 ITU-T 共同制订的 H.248 协议。在用户接入侧，通过相应接入网关和服务器的设置，不但可以接入移动终端，也可以接入固定电话终端、多媒体终端、PC 等，接入方式不限于蜂窝射频接口，也可以是无线的 WLAN、WiMAX，或者是有线的 LAN、DSL、同轴电缆等技术。因此，IMS 系统不但适用于移动网络，同样也适用于固定网络，这充分体现了固定/移动网络的融合。

同时，IMS 借鉴通信网的交换控制技术，在 IP 承载层之上引入了集中的呼叫/会话控制层。但是由于其下层已不是传统的电路交换网，而是 IP 分组网，因此 IMS 不再采用电信网信令，而是最大限度地重用 Internet 技术和协议。控制信令放弃使用 ITU-T 定义的在综合业务数字网(ISDN)信令基础上开发的成熟的 H.323 协议族，选用了虽然并未成熟然而在技术上却简单而又易于扩展的 SIP 协议。其核心网元呼叫会话控制功能(CSCF)相当于 SIP 代理服务器，除了完成 SIP 消息转发功能以外，还负责多媒体通信的呼叫控制。关于 SIP 消息的路由，对于 SIP 终端用户，重用域名系统(DNS)协议进行地址解析；对于使用传统电话号码的终端用户，重用 IETF 开发的电话号码映射(ENUM)协议进行地址解析。由于大部分网络接口采用的都是 Internet 协议，因此，IMS 系统不但支持 3G 用户之间基于 IP 网络的多媒体通信，而且也能毫无困难地支持 3G 用户和 Internet 用户之间的通信。

2）业务融合技术

业务融合是 IMS 更高层次的融合，也是对于未来异构信息通信网络更为重要的深层次的融合。从网络平滑演进的角度考虑，IMS 支持三种业务提供方式。

第一种方式是继续重用固定和移动智能网技术，由业务控制点(SCP)提供网络运营商的签约智能业务，以保证融合网络仍然能提供原来网络中的所有增值业务。其技术上的更新只是原来始终和交换机位于一体的业务交换功能(SSF)和 CSCF 分离，成为独立的 IM-SSF。

第二种方式是重用 SIP 网络的业务技术。由于 IMS 控制层采用的是 SIP 协议，因此很自然地可以通过设置各种 SIP 服务器，利用互联网中广泛使用的公共网关接口(CGI)、Servlet、JavaBean、呼叫处理语言(CPL)等技术，向用户提供各种业务应用。

第三种方式称为 Parlay/OSA 业务架构，即重用已经在软交换网络中应用的 Parlay 技术，通过和 Parlay 组织的密切合作，联合开发调用 IMS 网络能力的 API，实现基于 API 的开放式业务提供，由业务层的 OSA 应用服务器(AS)提供业务应用。这些应用服务器既可以属于网络运营商，也可以属于独立的第三方业务运营商，从而使通信网由封闭的网络成为开放的网络，使业务层和网络控制层真正分离。尤其是，3GPP 又针对常用的 API 定义了 Web Services 接口，使得 IMS 业务应用可以方便地和互联网中的 Web 服务相结合，实现通信网和 Internet 的业务融合。因此，Parlay/OSA 代表了 IMS 的发展方向，是业务融合的核心技术。

3）管理融合技术

IMS 在业务管理和运营管理上，充分重用电信网的运营支撑系统(OSS)和业务支撑系统(BSS)技术；在网络管理上借鉴 IP 网络接入认证技术和安全技术；在 QoS 控制和计费上提出了基于策略控制的 PCC 技术。

最后需要强调的一点是，IMS 的优势在于它是一个安全可靠、具有服务质量保证的

网络系统,这正是通信网区别于互联网的重要特点,因此,IMS 必须继续重视融合管理技术的研究,确保基于 IMS 的网络是一个可运营、可管理和可赢利的网络。

2.2.2 基于 SIP 的业务融合

在通信技术迅速发展的今天,电话网已经基本遍布全球。语音通信已成为人们日常生活中最基本的需求,它使得人们在任何时间、任何地点都能够进行无障碍的交流沟通。而随着通信业务量的增加,传统的电话网已经不能满足人们的需求。SIP 开发的目的是帮助提供跨越因特网的高级电话业务。IP 电话正在向一种正式的商业电话模式演进,SIP 就是用来确保这种演进实现的 NGN(下一代网络)系列协议中重要的一员,它能够帮助提供跨越 Internet 的高级电话业务,解决 IP 网中的信令控制以及与软交换机的通信,从而构成新一代的通信平台。

SIP (session initiation protocol)最早由 MMUSIC IETF 工作组在 1995 年研究,由 IETF 组织在 1999 年提议成为标准。SIP 是 IETF 标准进程的一部分,它是在诸如 SMTP(简单邮件传送协议)和 HTTP(超文本传送协议)基础上建立起来的。

SIP 不是会话描述协议,也不提供会议控制功能。SIP 应该和其他协议协同工作以提供给用户完全的服务,但是 SIP 的基本功能和操作并不依赖于任何其他协议。SIP 可以单独地工作,也可以与其他协议一起协同工作。

1. SIP 的体系结构

SIP 协议是一个 Client/Server 协议,RFC3261 中定义的 SIP 系统的逻辑实体包括用户代理(user agent,UA)、代理服务器(proxy)、注册服务器(registrar)、重定向服务器(redirect server)和 B2BUA(back-to-back user agent)。

1) 用户代理

UA 作为 SIP 的终端实体,分为两个部分:客户端 UAC(user agent client),负责生成和发起呼叫请求并处理对请求的响应;用户代理服务器 UAS(user agent server),负责接收呼叫请求并作出对该请求的响应。SIP 消息根据其所携带的不同的方法导致的操作也不完全相同。用户代理按照是否保存状态分为有状态代理、有部分状态代理和无状态代理。

2) 代理服务器

Proxy 是 SIP 系统的中间实体,负责接收用户代理发送来的请求,根据网络策略将请求发给相应的服务器,并根据收到的应答对用户作出响应,主要完成路由处理功能,保证了将请求发送到离目标用户更近的其他实体。

3) 注册服务器

注册服务器主要用于登记分组终端的当前位置和位置服务的原始数据。它会接收和处理用户端的注册请求,完成用户地址的注册。由于注册请求中 Contact 地址的有效期过短会引起注册刷新消息的频繁,从而给网络带来沉重负担;存亡周期过长则不利于运营商对用户终端的控制,因此对于注册请求中存亡周期过短或过长的行为,注册服务器应当能够进行正确的处理。

4）重定向服务器

引入重定向服务器是为了减轻代理服务器路由请求消息的负载，提高信令的可靠性。重定向服务器将一个请求的原地址映射为零个或多个地址，通过一个响应的形式再次返回给客户机。客户机根据此地址重新发送请求。当消息的请求方收到了重定向响应的时候，它将根据响应所给出的信息重新发送请求。

在结构上，重定向服务器由两层构成：服务器事务层和能够访问 location service 数据库的事务用户。location service 保存了单个 URI 和一组可选择的目标的映射关系，通过查找 location service 而获取目标 URI。

5）B2BUA

B2BUA 实质上是 SIP UA 的一种应用，是一种特殊的 SIP 逻辑实体，适用于 SIP 系统中需要呼叫和业务控制的场合。它可以接收 SIP 请求并像 UAS 那样处理它们。为了决定如何应答一个请求，B2BUA 又向别的实体发送请求，此时它扮演了 UAC 的角色。B2BUA 需要维护对话的状态，并处理所有在它所建立的对话中发送的请求消息。

原则上 B2BUA 不能改变 From 域和 To 域的 SIP URI 部分，以及其他有可能影响业务透明传输的消息组成部分。

上述功能实体在一个具体呼叫事件中扮演不同角色，而这样的角色不是固定不变的。例如，一个用户终端在会话建立时扮演 UAS，而在主动发起拆除连接时则扮演 UAC。SIP 的组网很灵活，可以根据具体情况定制。

2. SIP 的消息格式

SIP 协议是采用 UTF28 字符集来进行编码的文本协议，UTF8 的相关内容参见 RFC2279。SIP 协议消息分请求和响应两类，其中请求消息由客户机发往服务器，而响应消息则由服务器发往客户机。

除选用的字符集以及语法定义外，请求和响应消息均采用 RFC2822 定义的基本格式进行编码。请求和响应消息格式由一个起始行、若干个头字段以及一个可选的消息体组成。其中，消息体为可选项，头字段与消息体之间用空行进行分隔。值得注意的是，即使没有消息正文，也必须有空行[7]。

请求消息和响应消息都包括一个起始行（start-line）、一个或多个字段组成的消息头和作为可选项的消息体（message body）；SIP 消息头主要用来指明本消息是由谁发起和由谁接受，经过多少跳转等基本信息；消息体主要用来描述本次会话具体实现方式。其格式如下：

```
SIP 消息 =  起始行
     消息头部(1 个或多个头部)
     CRLF(空行)
     [消息体]
```

起始行 start-line＝request-line/status-line，消息头分为通用头（generic-header）、请求头（request-header）、响应头（response-header）和实体头（entity header）4 种。

请求消息的起始行为请求行（request-line）。请求行由方法名、请求 URL 和协议版本组成，各部分之间均用一个空格字符进行分隔。

响应消息的起始行为状态行(status-line),状态行以协议版本开始,然后用状态码和与状态码相关的文本描述,各个部分之间用一个空格字符SP进行分隔。

3. SIP 呼叫流程

SIP协议的组网模型图如图2-4所示。

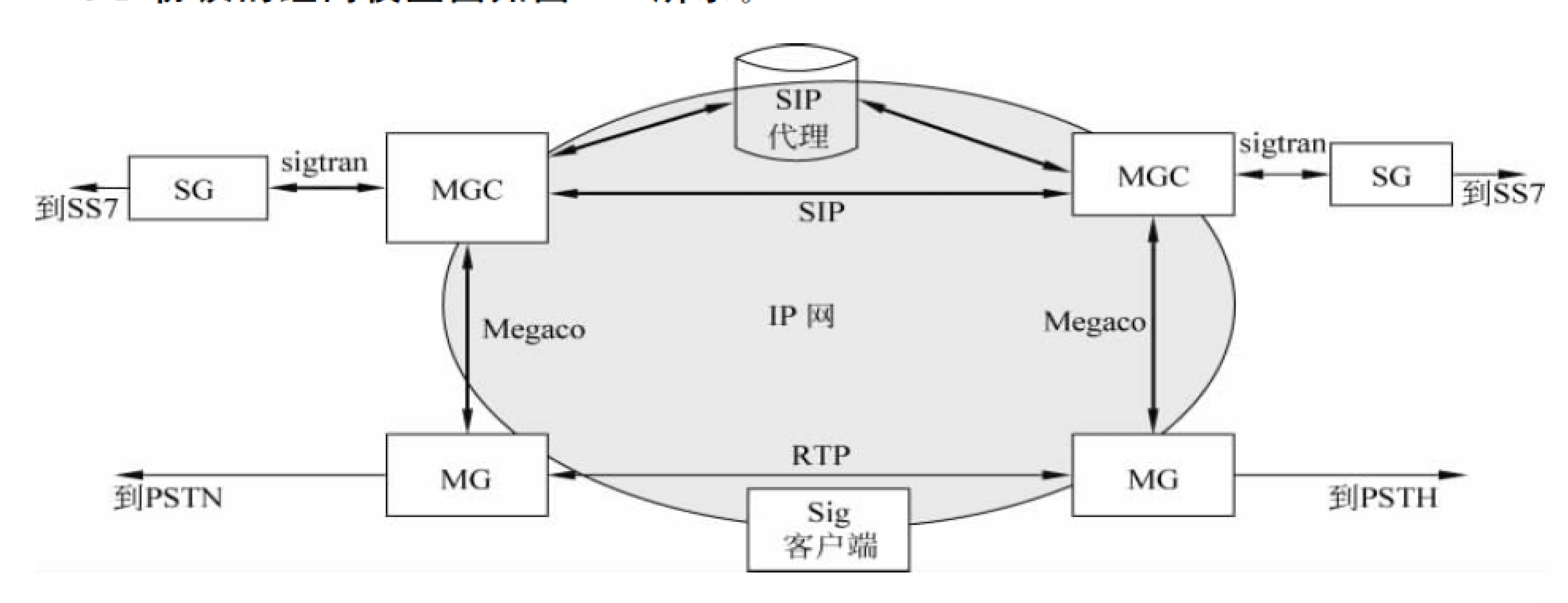


图2-4 SIP组网示例图

MGC(media gateway controller,媒体网关控制器)负责将PSTN前向信令映射为SIP请求,将SIP响应映射为PSTN后向信令,并且应该实现Megaco/h. 248协议以及SIP用户助理部分。SIP代理逻辑上是一个独立的实体,实现时可以作为单独的物理实体也可以与MGC捆绑在一起。负责转发或重定向SIP请求和响应。

SIP客户是一个单独的物理实体,可以是SIP电话、SIP会议终端等。MG(media gateway,媒体网关)负责PSTN域IP网之间媒体流的转换和转发。

MG之间、MG与SIP终端之间使用RTP/RTCP传输媒体流。

MG与MGC之间使用Megaco/h. 248协议或MGCP协议传输MGC对MG的控制信息以及MG向MGC上报的通知消息。

MGC与SIP代理、MGC与SIP客户或者SIP代理之间使用SIP协议来传输呼叫控制消息。

图2-5展示了一个PSTN用户呼叫另一个PSTN用户的流程。

流程说明:

(1) PSTN网络侧收到主叫用户发出的呼叫建立请求消息,生成初始地址消息IAM到软交换,请求路由。

(2) 软交换通过号码分析,不能够判别被叫用户为SIP用户,因此NNI接口上采用SIP-T信令。此时初始发送的IAM消息被组装成SIP INVITE请求消息发出,通过SIP系统的路由方式路由至代理服务器。

(3) 代理服务器将INVITE消息发送到被叫用户代理,即SIP终端。

(4) 代理服务器同时发送100 Trying响应给软交换,表明已经收到INVITE请求,呼叫建立请求正被转发至目的地,尚在进行中。

(5) SIP终端收到INVITE请求,向代理服务器发180 Ringing响应。

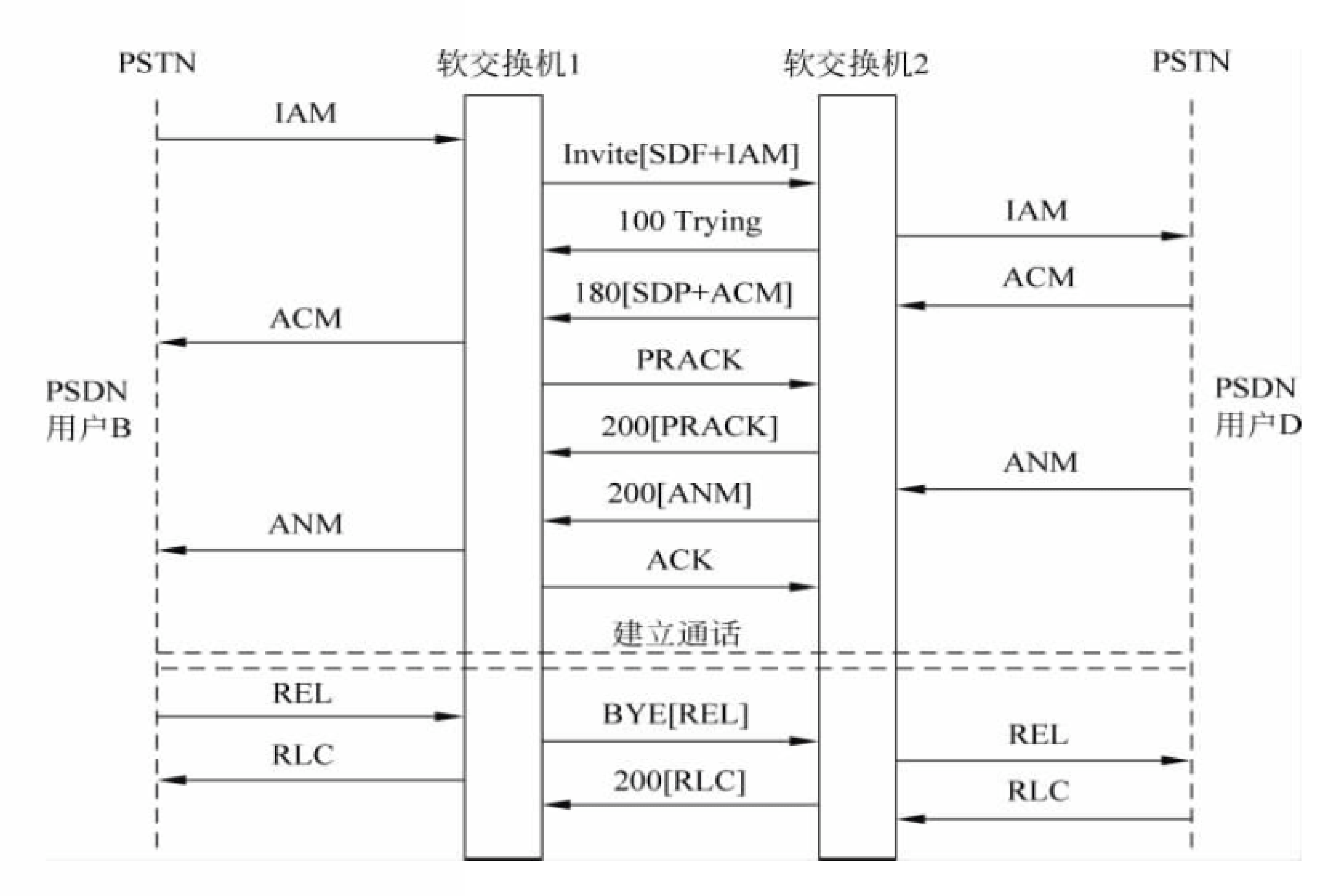


图 2-5　PSTN 用户呼叫 PSTN 用户

(6) 代理服务器将收到的 180 响应转给软交换。

(7) 软交换根据接收到的 180 消息,用 SIP-T 提取出 ACM 消息并结合本地策略,生成新的 ACM 消息,发送到主叫侧的 PSTN 网络。

(8) 被叫用户应答,SIP 终端向代理服务器发送 200 OK 响应。

(9) 代理服务器将 200 OK 响应转给软交换。

(10) 软交换接收到 200 消息后,提取出 ANM 消息并结合本地策略,发送到主叫侧的 PSTN 网络,网络负责通知主叫用户。

(11) 软交换同时发 ACK 给代理服务器。

(12) 代理服务器将 ACK 转给 SIP 终端,至此呼叫建立成功。

(13) 被叫用户间建立通话,进入通信阶段。

(14) 呼叫释放可由通信双方的任一方发起,假定由主叫方发出,PSTN 网络侧收到主叫方送出的释放请求消息,向软交换发送 REL 消息。

(15) 软交换接收到 REL 消息后,向主叫侧发送 RLC 消息;同时将 REL 消息封装在 BYE 消息中,发送到代理服务器。

(16) 代理服务器将 BYE 消息发给 SIP 终端。

(17) 代理服务器向软交换发送封装 RLC 的 200 OK 消息,同时接受被叫侧 PSTN 网络发送的 RLC 消息。

2.3　异构无线网络 IP 层的融合

下一代网络(NGN)的一个主要特点就是通用移动性,即指与接入技术无关的移动性,不论用户使用何种接入技术,都能向用户提供一致的服务。IP 技术恰恰可以屏蔽不

同类型接入网络的差异，为上层提供统一的接口。因此，基于IP的网络层移动性管理协议可以为终端在异构网络环境下的漫游提供统一的解决方案[8]。

传统的移动性管理技术与基于IP的网络层移动性管理技术区别主要在于：传统的移动通信系统的网络信令有专门的7号信令网来传输信令，信令传输与业务传输是相互独立的；而在全IP网络中，核心网和空中接口都已经全IP化，所有业务由IP直接承载，即控制和业务都在IP层面上同时承载。由于IP网络本身的性能和传统的7号信令网的差异，需要重新考虑IP平面上同时传送信令和业务数据时的性能要求。

下面就分别介绍网络层宏移动性管理技术——移动IPv4和移动IPv6，以及微移动性管理技术——分层移动IPv6。

2.3.1 移动IPv4

在IPv4网络中，对移动节点在不同IPv4子网之间的移动性进行管理，使移动节点在改变网络的接入点时，能保持通信的连续性。MIPv4的基本结构如图2-6所示。

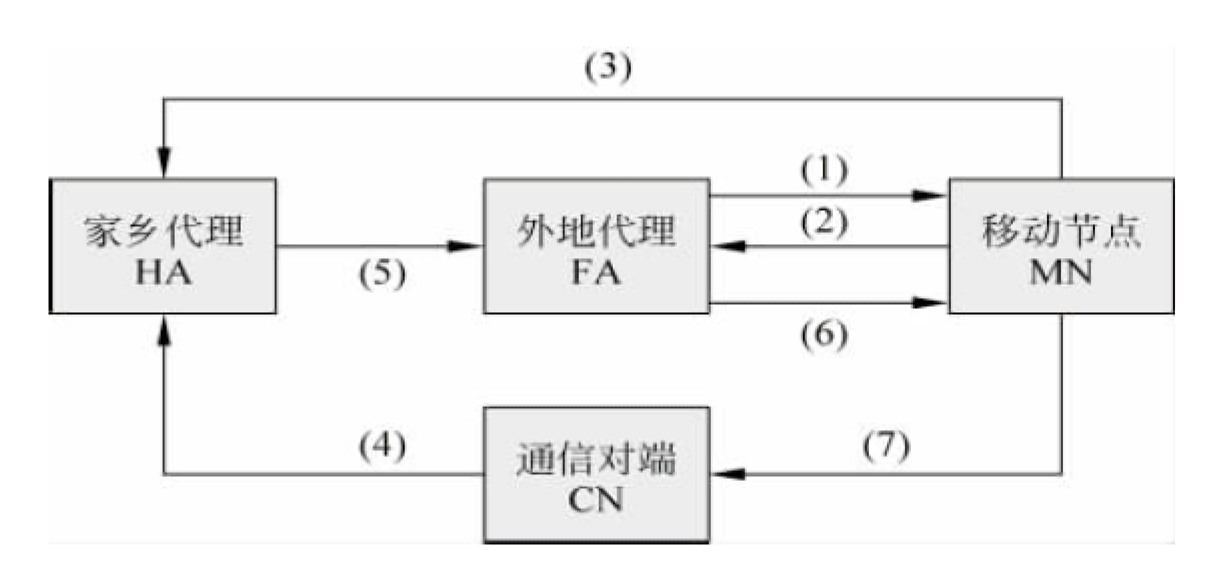


图2-6 MIPv4的基本结构

移动IPv4基本操作过程简要描述如下：

(1) 移动代理(HA和FA)通过代理通告消息告知移动节点(MN)移动代理的存在。若MN发现自己在家乡网络，则其操作和固定主机一样。若是从其他注册的网络回到家乡网络，将通过和家乡代理交换“注册请求”和“注册应答消息”在家乡代理上进行注销。

(2) 如果MN发现它已经移动到了一个外地网络上，它将获得该外地网络上的一个转交地址。这个转交地址可能来自外地代理的通告，也可能由DHCP等方式配置转交地址，前者称为外地代理转交地址，后者称为配置转交地址。

(3) 移动到外地网络上的MN随后与家乡代理交换注册请求和注册应答消息，注册它的转交地址，创建或修改移动绑定。

(4) HA截获发往移动节点家乡地址的数据分组。

(5) HA通过隧道把截获的数据分组发送到移动节点的转交地址。

(6) 隧道的输出端点(FA或MN本身)对收到的报文进行拆封后，交给移动节点。

(7) MN发出的报文通过标准IP路由机制被路由到目的节点，不需要经过家乡代理。

由上述过程可知：当MN移动到外地网络时，先通过代理搜索确定自己的位置和转

交地址，然后通过注册使代理完成转发功能，最后通过隧道技术完成数据包的路由选择。故移动 IPv4 的实现主要依赖于以下 4 种工作机制：代理发现机制、注册机制、隧道技术和路由选择。

1. 代理发现机制

移动 IPv4 实现代理发现的主要方法是，扩展 ICMP 路由器的发现机制，定义了“代理通告”和“代理请求”两个新的消息。移动代理周期性地发送代理通告消息，或者是移动主机主动发出代理请求消息。代理通告消息在 ICMP 路由器通告消息中增加“移动代理通告扩展”部分，代理请求消息除其 IP 包头的 TTL（生存时间）字段必须为 1 外，和 ICMP 路由器请求消息一致，如图 2-7 所示。



图 2-7　移动代理通告扩展格式

当移动节点收到代理通告消息后，判断自己是否从一个网络切换到另一个网络，是在家乡网络还是在外地网络，当切换到外地网络时，可以选择使用外地代理提供的转交地址。

在代理发现机制中，要求以下几点：不能被链路层协议发现的移动代理必须发送代理通告，能够被链路层协议发现的移动代理也应该实现代理通告；每个移动节点都必须实现代理请求，同时限制移动节点自己发送请求的速度；移动节点应该区分代理通告消息和作为其他用途的 ICMP 路由器通告消息；使用生存期字段的值或者网络前缀表示是否从一个子网移动到另外一个子网。

2. 注册机制

移动节点到达新的网络后，通过注册过程把新的可达性信息通知家乡代理，注册过程涉及移动节点、外地代理和家乡代理，通过交换注册消息，在家乡代理上创建或者修改“移动绑定”，使家乡代理在规定的生存期内保持移动节点家乡地址和转交地址的关联。

注册有两种过程：一是通过外地代理转发移动节点的注册请求；另一种是移动节点直接到家乡代理上进行注册请求[9]。具体过程分别如图 2-8 和图 2-9 所示。

这种注册过程需要下面 4 个消息：

(1) 移动节点发送注册请求到预期的外地代理，开始注册过程。

(2) 外地代理处理注册请求，然后把它转发到家乡代理。

(3) 家乡代理发送注册应答到外地代理，同意或者拒绝这个请求。

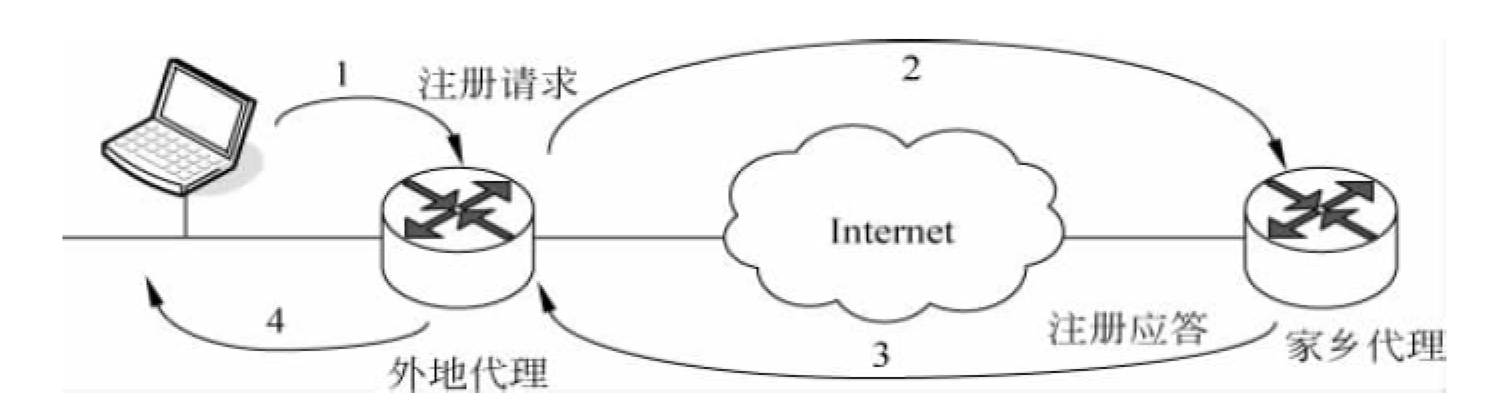


图 2-8 经过外地代理转发的注册过程

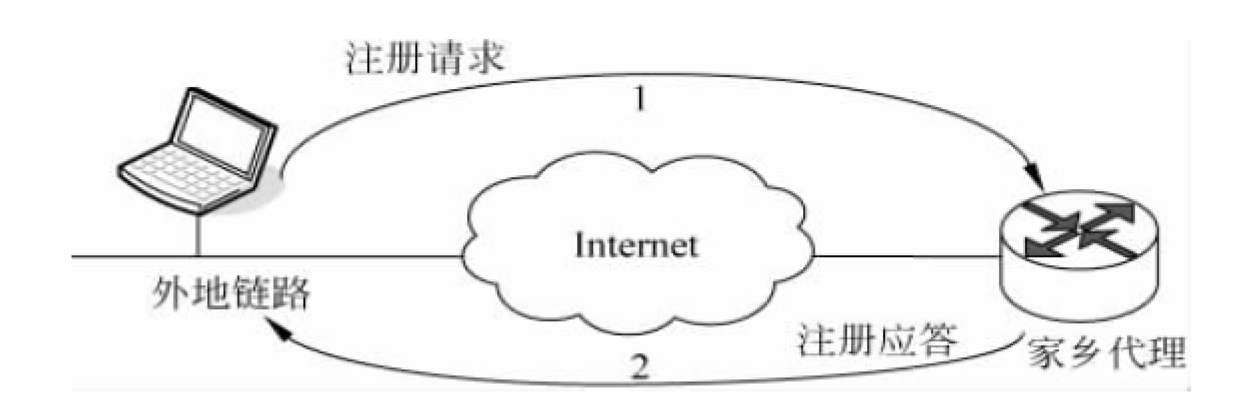


图 2-9 不经过外地代理转发的注册过程

(4) 外地代理处理这个应答,把处理的结果告知移动节点。

移动节点直接到家乡代理上进行注册请求的过程需要下面两个消息:

(1) 移动节点发送注册请求给家乡代理。

(2) 家乡代理给移动节点发送一个注册应答,同意或者拒绝这个请求。

如果在一个确定的时间间隔没有收到任何注册应答,MN 可以再发送一个注册请求,并使用时间戳为每次重传选择一个新的注册标识,进行一次新的注册。

3. 隧道技术

隧道技术是一种改变 IP 数据分组传输路由的方法,例如,将数据分组传输到使用通常的路由方法不能到达的某个节点。IETF 工作组定义了三种 IP 封装的类型:IP 的 IP 封装、最小封装和通用路由封装。

1) IP 的 IP 封装(IP in IP encapsulation)

这是一个因特网标准,用于将整个 IPv4 数据包放在另一个 IPv4 数据包的净负荷部分。新 IP 分组包头的关键字段应设置如下:

源地址和目的地址分别设为隧道的入口点和出口点;协议类型字段设为 4,表示净负荷本身也是 IP 分组(包括包头和净负荷);IP 的 IP 封装可以在任何情况下使用,无论 IP 包是否进行了分片。

IP 的 IP 封装如图 2-10 所示。

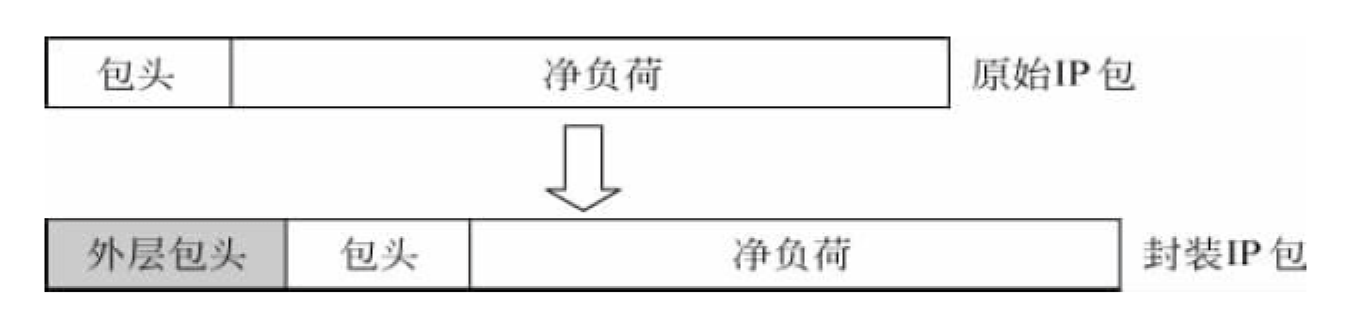


图 2-10 IP 的 IP 封装

2）最小封装(minimal encapsulation)

最小封装的目的是减少实现隧道所需的额外字节数，可通过去掉IP的IP封装中的内层IP包头和外层IP包头的冗余部分来完成。

最小封装与IP的IP封装相比可以节省一些字节，但也带来了一些不便之处：首先，最小封装不能用于那些已经经过分片的原始数据包；其次，在隧道内的每一台路由器上，原始包的生存时间域的值都会被减小；最后，最小封装的包中可能不包含原始源地址域，从而不能保证隧道内的ICMP报文可以到达原始包的源。IP的最小封装如图2-11所示。

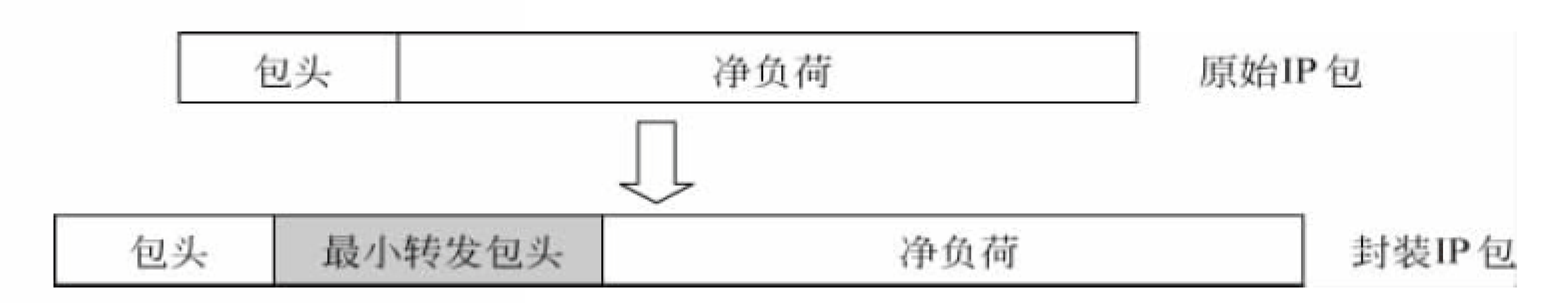


图2-11　IP的最小封装

3）通用路由封装(generic routing encapsulation，GRE)

GRE除了支持IP，还支持其他的网络层协议。它允许一种协议的数据包封装在另一种协议的数据包的净负荷中。GRE的封装过程如图2-12所示。



图2-12　通用路由封装

4．路由选择

当移动节点在外地链路上收包时，首先家乡代理广播对家乡地址网络前缀的可到达性，然后由家乡代理截获数据包。如果家乡代理是多端口路由器，则广播对移动节点家乡地址的可到达性。家乡代理通过免费ARP，代理ARP，用家乡代理的MAC地址代替移动节点的MAC地址。移动节点回到家乡链路后，主动发送免费ARP，恢复自己的MAC地址与IP的对应关系。之后家乡代理向移动节点隧道封装传递分组。

当移动节点在外地链路上发包时，首先把包发到外地代理，然后按照常规IP路由协议路由到目的地址。

当通信对端向移动节点发数据包时，存在三角路由问题。由通信对端送给连接在外地链路上的移动节点的数据包先被路由到它的家乡代理上，然后经过隧道送到移动节点的转交地址，然而，由移动节点发出的数据包却直接路由到了通信对端，这构成了一个三角形，就是“三角路由”问题。

2.3.2　移动IPv6

IPv4协议是一个简单而又有效的网络互联协议，能够连接少至几个节点，多至

Internet 上多个主机。然而,随着 Internet 的发展,IPv4 的一些缺陷也逐渐暴露出来,因此,需要对 IPv4 进行升级和完善。为了从根本上解决在 IPv4 网络中遇到的各种问题,引入了移动 IPv6。IPv6 的基本操作过程如图 2-13 所示。

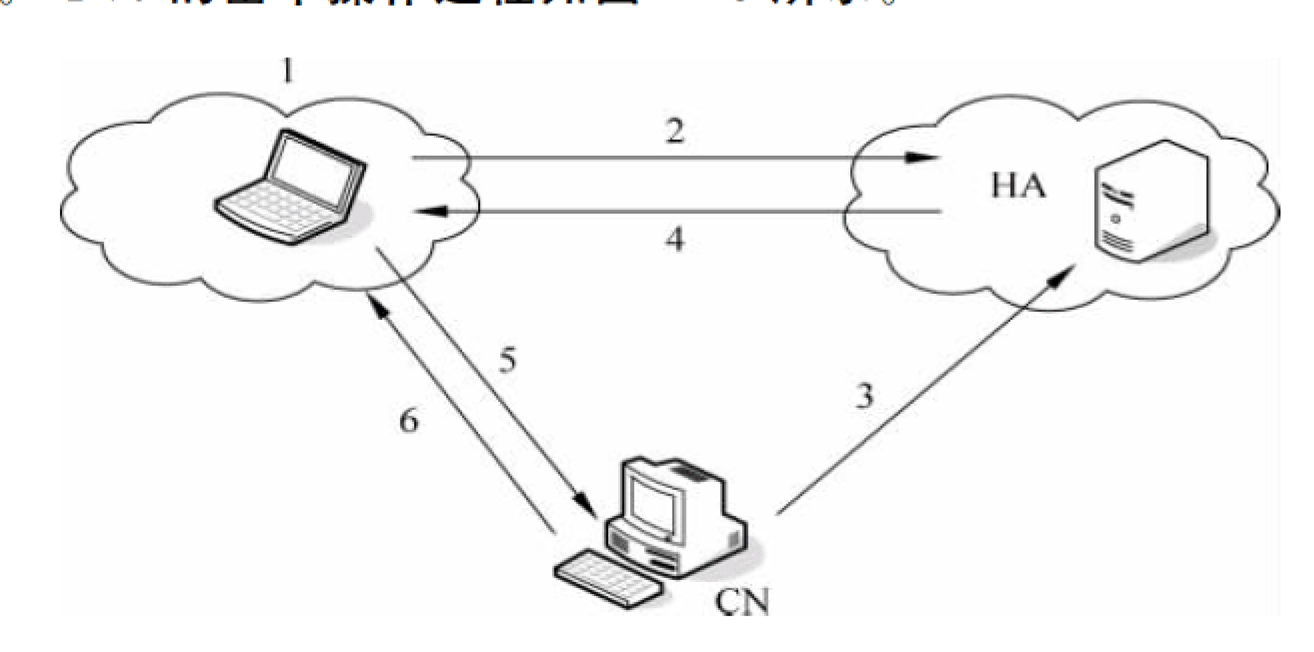


图 2-13　IPv6 的基本操作过程

(1) 当移动节点 MN 收到"邻居发现机制"发送的通告,确认自己已经连接到外地链路以后,通过自动配置获得一个转交地址。

(2) MN 向家乡代理申请注册,为移动节点的家乡地址和转交地址在家乡代理上建立"绑定",使得家乡代理能够把只知道其家乡地址的节点发来的分组转发到移动节点的当前位置。

(3) 一个通信对端想要与移动节点进行通信,则发送分组到移动节点的家乡地址。

(4) 家乡代理获得这个分组,转发给 MN。

(5) MN 检测到家乡目的地址,则执行与通信对端之间的绑定更新。

(6) 在绑定更新确认以后,MN 和 CN 之间能够直接通信,不需要 HA 的转发,这就实现了路由优化,避免了"三角路由"问题。

移动 IPv6 在通信中经历以下几个过程:

(1) 移动检测过程。MIPv6 草案中定义的移动检测机制主要是利用 IPv6 的邻居发现机制(包括路由发现和邻居不可达检测),即 MN 通过检测当前默认路由器是否双向可达来判断自己是否发生了移动。如果当前默认路由器已经双向不可达,而且发现了一个新的默认路由器,MN 就可以假定自己已经移动到了另一条链路上。

(2) 转交地址的形成过程。MIPv6 允许 MN 同时拥有多个转交地址,其中一个称为主转交地址。MN 在检测到移动发生后,应该使用新的默认路由器提供的一个子网前缀形成新的主转交地址。新的转交地址生成后,可以对这个地址进行重复地址检测(duplicated address detection,DAD),以确认它的唯一性。

(3) 动态家乡代理地址发现。当离开家乡的 MN 进行 HA 注册时,可能不知道家乡链路上的哪些路由器提供 HA 服务。这种情况下,MN 可以使用动态 HA 发现机制去获取家乡链路上的家乡地址。该机制使用 ICMP 家乡代理地址发现请求和应答两种消息。

首先,MN 发送 ICMP 家乡代理地址发现消息到其家乡网络前缀所对应的 MIPv6 家乡代理泛播地址。接着,收到此消息的 HA,返回给 MN 一个相应的应答消息。消息中给出了家乡链路上所有 HA 的地址列表,按 HA 的优先值降序排列,优先值相同的则随

机排列。最后,MN 从列表中选择优先值最高的 HA 进行注册。

(4) 绑定更新。当 MN 收到“邻居发现机制”发送的通告,确认自己已经连接到外地链路以后,通过常规的 IPv6 无状态或有状态的自动配置机制,获得一个转交地址。转交地址的子网前缀是移动节点正访问的外地链路的子网前缀。

MN 在外地网络获得转交地址后,需要向家乡代理申请注册,为移动节点的家乡地址和转交地址在家乡代理上建立“绑定”,使得家乡代理能够把只知道其家乡地址的节点发来的分组转发到移动节点的当前位置。此过程称为“家乡注册”,如图 2-14 所示。

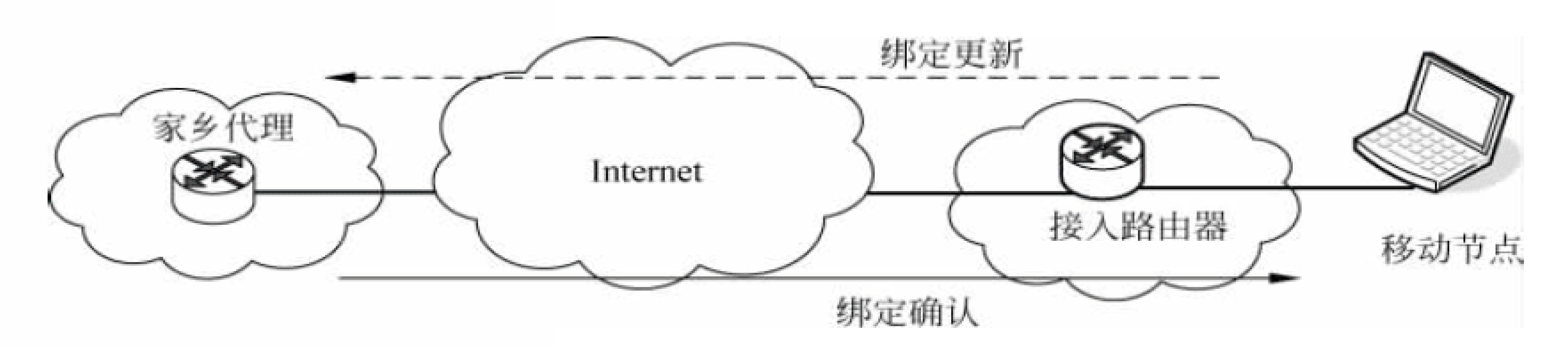


图 2-14　移动 IPv6 家乡注册过程

当 MN 向 HA 进行注册以后,移动节点和通信对端(CN)就可以通过 HA 进行通信了,但是这会导致“三角路由”问题。为了避免这个问题,采用“优化路由”,即让 MN 和 CN 直接通信,而不需要 HA 的转发。这就需要在 MN 和 CN 之间进行一种绑定,也就是进行“返回路径可达过程(RRP)”。通过返回路径可达过程,CN 知道自己是否能够使用转交地址和家乡地址访问移动节点。如果此 RRP 过程测试失败,CN 将既不能接受移动节点的绑定更新,也不能直接发送分组到移动节点的转交地址。

相比于 IPv4 协议而言,IPv6 协议的变化主要体现在以下几个方面[10]:

(1) 修改地址体系结构。地址长度扩展为 128 位,地址空间足够给地球上现在和未来的每一个网络设备分配一个 IP 地址。地址类型分为单播地址、组播地址和泛播地址。根据地址的有效范围,地址可以分为全球地址、局域地址和链路地址。

(2) 修改分组头格式。IPv6 的分组头固定长度是 40 个字节,只有 8 个字段,比起 IPv4 分组头至少包含 12 个字段要节省得多。同时,IPv6 中不再有校验和字段,分段字段移到了扩展包头中,简化了路由器的处理。

扩展包头的设计,既保证了实现选项所需的功能,又可以减少中间路由器处理的复杂度。另外,每个扩展包头还可以包含多个子选项,可以自由扩展。IPv6 协议既简单又灵活,具有良好的可扩展性。

(3) IPv6 实现了“流”的概念。“流”是指从一个特定源发向一个特定目的地的分组序列。

(4) IPv6 在安全方面做了重大的改进,将 IPSec 集成到了 IPv6 的标准中。

总之,IPv6 巨大的地址空间意味着 IP 可以继续增长而无须考虑资源的匮乏;对于分组包头的简化减少了路由器上所需的处理过程,从而提高了选路的效率;改进对扩展包头和选路的支持意味着可以在几乎不影响普通数据分组和特殊分组选路的前提下适应更多的特殊需求;流标记办法为更加高效地处理数据流提供了一种有效机制,有利于改善实时应用的服务质量;而安全性的改进,使得 IPv6 更加适用于那些要求对敏感信息和资源加以保护的商业应用。

2.3.3 分层移动 IPv6

分层移动 IPv6（HMIPv6）是一个微移动性管理模型，其目的是减少 CN 与 HA 之间的信令数量，改善移动 IP 的切换速度。HMIPv6 在 MIPv6 的基础上，引入了一个新的实体，称为移动性锚点（MAP），通过 MAP 管理本地切换，而全球移动性仍由 MIPv6 协议管理。HMIPv6 中定义了三个新术语：区域转交地址 RCoA、链路转交地址 LCoA 和本地绑定更新 LBU[11]。RCoA 是 MN 收到“MAP 选项”时根据 MAP 子网前缀自动配置的一个转交地址，也是 MN 向 HA 和 CN 注册的地址。LCoA 是 MN 根据路由器公告消息 RA 所配置的转交地址，是被 MAP 用来标识 MN 在本地域内不同子网中切换的地址，即相当于 MIPv6 中的 CoA。

HMIPv6 的主要原理是，移动节点在 HA 登记 MAP 的转交地址，因此，当移动节点在本地移动（即它的 MAP 不变）时，只需要在当前 MAP 上登记它的新位置，而不需要与 HA 或接入网之外的任何 CN 进行有关操作。通过使用这种方法，信令只发生在较小区域，不会扩展到核心网，完成位置更新的时间较短。在 HMIPv6 中，MAP 拦截所有发往已向它注册的 MN 的包，然后通过隧道发往对应的 LCoA。进行 MAP 发现可以使用动态 MAP 发现和路由器重编号 MAP 发现两种方法。

与 MIPv6 类似，HMIPv6 依赖于底层接入技术，允许在不同类型的接入网络间进行漫游。MN 进入一个 MAP 域会收到包含一个或多个本地 MAP 信息的路由器宣告。MN 将当前地址 LCoA 和属于 MAP 子网的一个 RCoA 地址绑定。MAP 接收所有已向它注册的 MN 的数据包，并把发往这些 MN 的包通过隧道直接发送到它们的 LCoA 上。MN 在同一 MAP 域内改变 LCoA 地址时，它只需向 MAP 注册新的 LCoA 地址。因此，只有 RCoA 改变时才需要向 HA 和 CN 注册。MN 在同一 MAP 域内改变 LCoA 地址时，它的 RCoA 地址不变，这使得它的位置对于正在与之通信的 CN 是透明的。

HMIPv6 的移动性管理包含以下两种。

1）MAP 域内移动性管理

在 HMIPv6 中，如果设置了 MAP 选项中的 I 和 P 标志位，MN 可以不需要使用家乡地址选项而把 RCoA 作为源地址来发送包。它也可以把 RCoA 作为源地址来向 MAP 发送本地绑定更新。当 MN 在同一个 MAP 域中移动时，在 CN 看来它就像一个固定的节点一样。

2）MAP 域间移动性管理

当 MN 在不同 MAP 之间漫游时，采用基本的移动 IP 宏移动性管理。

最后介绍一下 HMIPv6 的主要工作机制。当一个支持 HMIPv6 的 MN 接收到路由器宣告时，它会从中查找 MAP 选项。该选项可能包含一个或者多个不同 IP 地址或子网前缀。MN 应该优先向最高优先级的 MAP 进行注册。优先级为零的 MAP 不能接入新的绑定。当接收到多个优先级不为零的 MAP 消息时，MN 可以根据“距离”域的值来选择一个 MAP 进行注册。

MAP 选项包含的生存时间值为零表示 MN 不能选择该 MAP。生存时间值为零说明 MAP 不工作。当接收到该选项时，MN 必须选择另一个 MAP 并建立新的绑定。所

有在该 MAP 的绑定都会丢失。

当 MN 进入多个 AR 的重叠区时，它应该使用属于它当前 MAP 的 AR 所在子网的 LCoA。

MN 必须保存接收到的选项并至少选择一个 MAP 进行注册。这些被保存的选项很重要，因为移动检测算法需要将它们与其他后来接收的选项进行比较。

在进行 MAP 注册时，如果 R 标志被置位，则 MN 必须使用 RCoA 代替它的家乡地址。然后它的 RCoA 和 LCoA 的绑定被保存在 MAP 的绑定缓存中。如果 I 标志被置位，则 MN 将根据是否需要隐藏位置来选择是否在外发包中使用 RCoA 作为它的源地址。如果 P 标志被置位，MN 必须使用 RCoA 作为源地址。原因是网络管理者不想向外部网络暴露某些地址前缀的需要。V 标志指示如果 MN 使用 RCoA 作为源地址，则它必须将所有外发包从隧道发送到 MAP。该标志只有在 P 或 I 标志被置位时才有效。这个标志的目的是解决 AR 的入口滤波器可能不允许接入 RCoA 前缀的问题。如果 P 和 I 标志被置位，MN 可以同时选择一个以上的 MAP 或同时使用 MAP 地址(RCoA)和本身的地址(LCoA)来与不同的 CN 进行通信。

2.4 无线网络物理层融合

在异构网络的物理层上，近些年来出现了多种热点技术。本节重点介绍 4 种物理层关键技术：多天线技术、OFDM 技术、分集接收技术和自适应编码调制技术。

2.4.1 多天线技术

最近 10 年来，因特网和移动通信飞速发展，在第三代蜂窝移动通信中已经部分地引入了无线因特网和多媒体业务。而在新一代移动通信系统(即 LTE 和 5G)中，人们对传输速率提出了更高的要求，这就需要采用更先进的技术来实现更高的传输速率。然而频谱资源总是有限的，要支持高速率就要开发具有极高频谱利用率的无线通信技术。多天线技术在提高频谱效率、支持更高速的数据传输、提高传输信号质量、增加系统覆盖范围和解决热点地区的高容量要求等方面有无可比拟的优势，已经成为目前研究的热点问题，并广泛应用于各种移动通信系统中。

1. MIMO 技术简介

MIMO(multiple input multiple output)系统框图如图 2-15 所示。该技术最早是由 Marconi 于 1908 年提出的，它利用多天线来抑制信道衰落。

MIMO 技术是指在发送端和接收端分别设置多副发送天线和接收天线，其出发点是将多发送天线与多接收天线相结合以改善每个用户的通信质量(如差错率)或提高通信效率(如数据速率)。MIMO 技术实质上是为系统提供空间复用增益和空间分集增益，空间复用可以大大提高信道容量，而空间分集则可以提高信道的可靠性，降低信道误码率等。

通常，多径会引起衰落，因而被视为有害因素，然而对于 MIMO 来说，多径可以作为

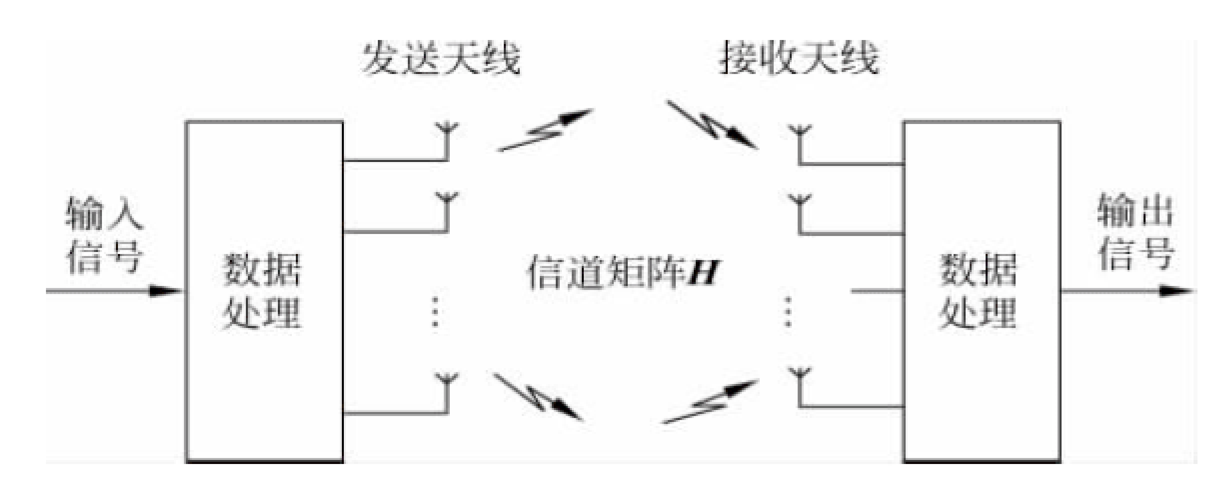


图 2-15 MIMO 系统框图(N 个发送天线,M 个接收天线)

一个有利因素加以利用,MIMO 技术的关键是能够将传统通信系统中存在的多径衰落影响因素变成对用户通信性能有利的增强因素,MIMO 技术有效地利用随机衰落和可能存在的多径传播来成倍地提高业务传输速率,因此它能够在不增加所占用的信号带宽的前提下使无线通信的性能改善几个数量级。假定发送端有 N 个发送天线,M 个接收天线,在收发天线之间形成 $M\times N$ 信道矩阵 $\boldsymbol{H}$,在某一时刻 t,信道矩阵为

$$\begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1N} \\ h_{21} & h_{22} & \cdots & h_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ h_{M1} & h_{M2} & \cdots & h_{MN} \end{bmatrix} \tag{2-1}$$

其中,$\boldsymbol{H}$ 的元素是任意一对收发天线之间的增益。对于信道矩阵参数确定的 MIMO 信道,假定发送端不知道信道信息,总的发送功率为 ρ,与发送天线的数量 M 无关;接收端的噪声用 $N\times 1$ 向量 n 表示,是独立零均值高斯复变量,各个接收天线的噪声功率均为 σ^2;发送功率平均分配到每一个发送天线上,则容量公式为

$$C = \log_2\left[\det\left(I_M + \frac{\rho}{N}HH^H\right)\right]\text{b/s/Hz} \tag{2-2}$$

令 M 不变,增大 N,使得 $\frac{1}{N}HH^H\to I_M$,这时可以得到容量的近似表达式为

$$C = \log_2(1+\rho)\text{b/s/Hz} \tag{2-3}$$

从上式可以看出,此时的信道容量随着天线数量的增大而线性增大。也就是说,可以利用 MIMO 信道成倍地提高无线信道容量,在不增加带宽和天线发送功率的情况下,频谱利用率可以成倍地提高。同时,利用 MIMO 技术可以成倍提高信道的可靠性,降低误码率。前者是利用 MIMO 信道提供的空间复用增益,后者是利用 MIMO 信道提供的空间分集增益。目前 MIMO 技术领域另一个研究热点就是空时编码。常见的空时编码有空时块码、空时格码。空时编码的主要思想是利用空间和时间上的编码实现一定的空间分集和时间分集,从而降低信道误码率。

2. 多天线技术的应用模式

频率资源的使用是有限的,无论在时域、频域还是码域上处理信道容量均不会超过香农极限。多天线的使用使得不同用户的信号可以用不同的空间特征来表征,使得空域资源的使用成为可能。空域处理可以在不增加带宽的情况下成倍地提升信道容量,也可以改善通信质量、提高链路的传输可靠性。

未来的多天线技术应用模式必将是灵活多变的，主要的多天线应用模式包括以下几种。

1）接收分集（单输入多输出时）

由于部分终端受尺寸大小、发射功率和成本等的影响，通常在发送端只有一根天线，基站使用多根接收天线，实现接收分集，理想情况下可获得 $10\log N_r$(dB)的增益，N_r 为基站接收天线的个数。容量随着接收天线的个数对数增加。应用场景如图 2-16 所示。

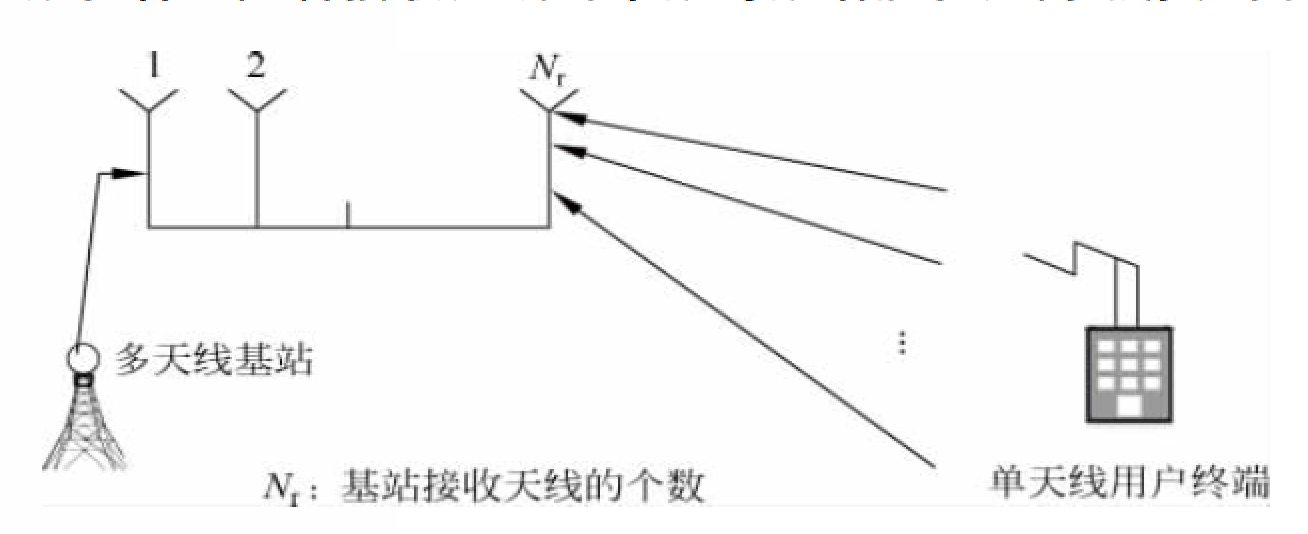


图 2-16　单输入多输出示意图

2）发送分集（多输入单输出时）

终端一根接收天线，基站多根发送天线，理想情况下可获得 $10\log N_t$(dB)的增益，N_t 为基站发送天线的个数。容量随着发送天线的个数对数增加。应用场景如图 2-17 所示。

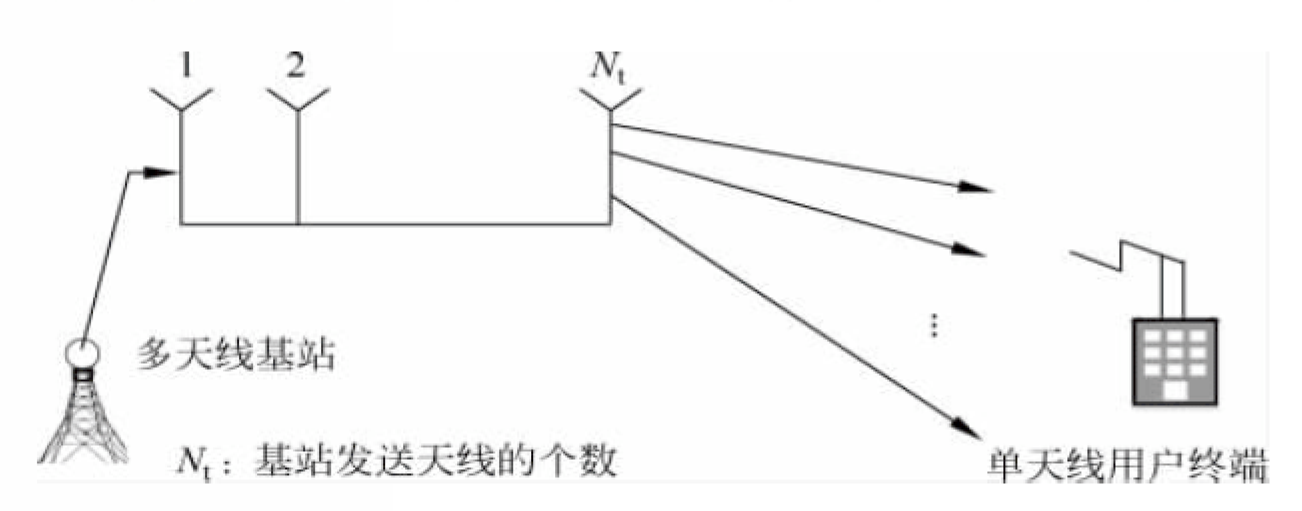


图 2-17　多输入单输出示意图

3）波束形成（多输入单输出时）

终端只有一根天线，基站使用多根发送天线，实现波束形成，由于在发送端已经得到了 $\boldsymbol{H}$ 矩阵，波束形成比发送分集信噪比提高 3dB。必须经过上行测量或者上行反馈获取信道信息，才能够进行波束形成。

4）空时编码（多输入多输出时）

未来的通信系统中，终端会走向多样化，部分终端可以拥有多根天线，这样通信链路的上下行均可实现多输入多输出（MIMO），示意图如图 2-18 所示。空时编码是 MIMO 的主要应用形式之一，正交的空时分组编码可以获得满分集增益，空时网格编码不仅能够获得部分的分集增益，同时也能够获得编码增益。

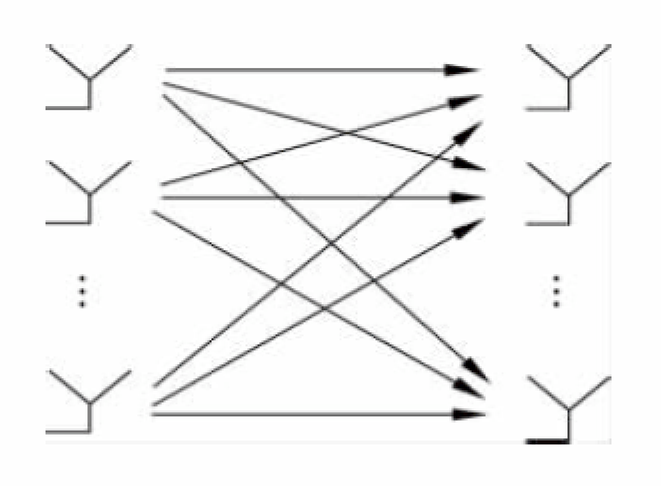


图 2-18　多输入多输出示意图

5）空间复用（多输入多输出时）

MIMO 的另一种主要的应用形式是空间复用。空间复用技术使得信道容量成倍地增长变为可能。使用空间

复用技术必须满足：$N_r \geqslant N_t$，使用迫零和干扰对消进行逐符号检测，发送端无须知道信道信息，无须通道校正，当信道容量下降时，复用系数应该自适应改变。

6）智能天线（先进的多天线系统）

智能天线的一个主要的任务是如何获取和利用信号的空间方向信息，并通过阵列信号处理改善信号的质量，从而提高系统的性能。天线阵列的加权在基带通过数字信号处理完成，自适应阵列技术属于其中的一部分。自适应天线阵列是智能天线技术的研究重点和发展方向。

3. 多天线技术的空域自适应

未来的多天线技术必将实现空域自适应链路。根据信道的变化，可以实现目标为最大的数据传输速率的链路自适应和平均信道容量最大的链路自适应。

实现目标为最大的数据传输速率的链路自适应的设计原则：

（1）移动环境下的 MIMO 信道是变化的，容量也是变化的。

（2）在低秩信道下并非发送天线越多信道容量越大，可以通过合理地选择发送天线来提升系统容量。

实现目标为平均信道容量最大的链路自适应的设计原则：

（1）当收发天线之间的衰落系数互不相关且服从相同的分布时，MIMO 系统将获得可观的信道容量。但是由于阵元间距和实际通信环境所限，各对收发天线间的衰落系数往往是相关的。研究表明，在相关性较强的情况下，信道容量会大幅降低。

（2）在相关衰落信道中应该合理设计天线阵间距和排布方式来尽量降低阵元之间信道响应的相关系数。

4. MIMO 技术的应用

目前，3GPP 在高速下行分组接入方案（HSDPA）中提出了使用 MIMO 天线系统，这种系统在发送和接收方都有多副天线，可以认为是双天线分集的进一步扩展。另外，在 3GPP 的 WCDMA 协议中，涉及 6 种分集发射方法：空时分集发射（space time transmit diversity，STTD）、时间切换分集发射（time-switch transmit diversity，TSTD）、两种闭环分集发射模式、软切换中的宏分集，以及站点选择分集发射（site selection diversity transmit，SSDT）。宏分集是指在 CDMA 系统的软切换过程中，可以通过两个甚至三个基站同时向一个移动台发射同样的信号，这是宏分集发射；同样，接收时通过相邻的基站进行分集接收（多个基站接收），即进行宏分集接收。而在 LTE 系统中，MIMO 技术包括三种，分别是发分集、波束成形和空间复用。

MIMO 可以应用于所有的无线通信技术。在 WiMAX802.16e 系统中，MIMO 和 OFDMA（orthogonal frequency division multiple access，正交频分复用接入）的完美结合，更能体现出 MIMO 的技术优势。

MIMO 系统可以抗多径衰落，但对于频率选择性衰落，MIMO 仍然无能为力，其他通信系统一般采用均衡技术来解决 MIMO 系统中的频率选择性衰落问题。WiMAX 的 OFDMA 技术可以很好地克服频率选择性衰落。下一代移动通信需要高频谱利用率的技术，但 OFDMA 提高频谱利用率的能力毕竟有限。结合 MIMO 技术，可以在不增加系

统带宽的情况下进一步提高频谱效率。MIMO+OFDMA 技术既可以提供更高的数据传输速率，又可以通过分集达到很强的可靠性和增强系统的稳定性。另外，OFDMA 由于码率低和加入了时间保护间隔而具有很强的抗多径干扰能力，多径时延小于保护间隔使系统不受码间干扰的影响，这样就可以使单频网络使用宽带 OFDMA 系统依靠 MIMO 技术消除阴影效应，真正实现网络的无缝覆盖。

2.4.2 OFDM 技术

在传统的多载波通信系统中，整个系统频带被划分为若干个互相分离的子信道，也就是所谓的载波。为了避免信道之间的干扰，在信道之间通常有一定宽度的保护间隔，接收端通过滤波器把各个子信道分离之后接收所需信息。这样虽然可以避免不同信道的互相干扰，但却以牺牲频率利用率为代价。而且当子信道数量很大的时候，大量分离各子信道信号的滤波器的设置就成了几乎不可能的事情。

20 世纪中期，人们提出了频带混叠的多载波通信方案，选择相互之间正交的载波频率作为子载波，也就是所说的正交频分复用(orthogonal frequency division multiplexing，OFDM)技术。OFDM 是一种无线环境下的高速传输技术。主要是在频域内将所给信道分成许多正交子信道，在每个子信道上使用一个子载波进行调制，且各个子载波并行传输。OFDM 特别适合于在存在多径传播和多普勒频移的无线移动信道中传输高速数据。OFDM 能有效对抗多径效应，消除 ISI，对抗频率选择性衰落，信道利用率高。

用户的信息首先要经过串行到并行的转换，转变成多个低速率的数据码流，通过编码之后，调制为射频信号，然后在正交的频率上并行地传送多路信号，这样 OFDM 能够充分地利用信道的带宽。OFDM 不用带通滤波器来分隔子载波，而是通过快速傅里叶变换(FFT)来选用那些即便混叠也能够保持正交的波形。

OFDM 尽管还是一种频分复用(FDM)，但已完全不同于过去。OFDM 的接收机实际上是通过 FFT 实现的一组解调器。它将不同载波搬移至零频，然后在一个码元周期内积分，其他载波信号由于与所积分的信号正交，因此不会对信息的提取产生影响。OFDM 的数据传输速率也与子载波的数量有关。

OFDM 系统的子载波可以自适应地根据信道的情况选择调制方式，并且能够实现在各种调制方式之间的切换。选择和切换的原则是频谱利用率和误码率之间的平衡选择。在通常的通信系统中，为了保持一定的可靠性，选择通过采用功率控制和自适应调制协调工作的技术。信道好的时候，发射功率不变，可以增强调制方式(如 64QAM)，或者在低调制(如 QPSK)时降低发射功率。功率控制与自适应调制要取得平衡，也就是说对于一个远端发射台，它有良好的信道，若发送功率保持不变，可使用较高的调制方案如 64QAM；若功率可以减小，调制方案也相应降低，可使用 QPSK。

1. OFDM 系统结构与收/发原理

移动通信信道的突出特点之一就是信道存在多径时延扩展，它限制了数据速率的提高，因为如果数据速率高于信道的相干带宽，信号将产生严重失真，信号传输质量大幅度下降。而 OFDM 技术由于具备上述特点，是对高速数据传输的一种潜在的解决方案，因

此，OFDM 技术已基本被公认为 4G 的核心技术之一。OFDM 系统结构框图如图 2-19 所示，其核心是一对傅里叶变换。

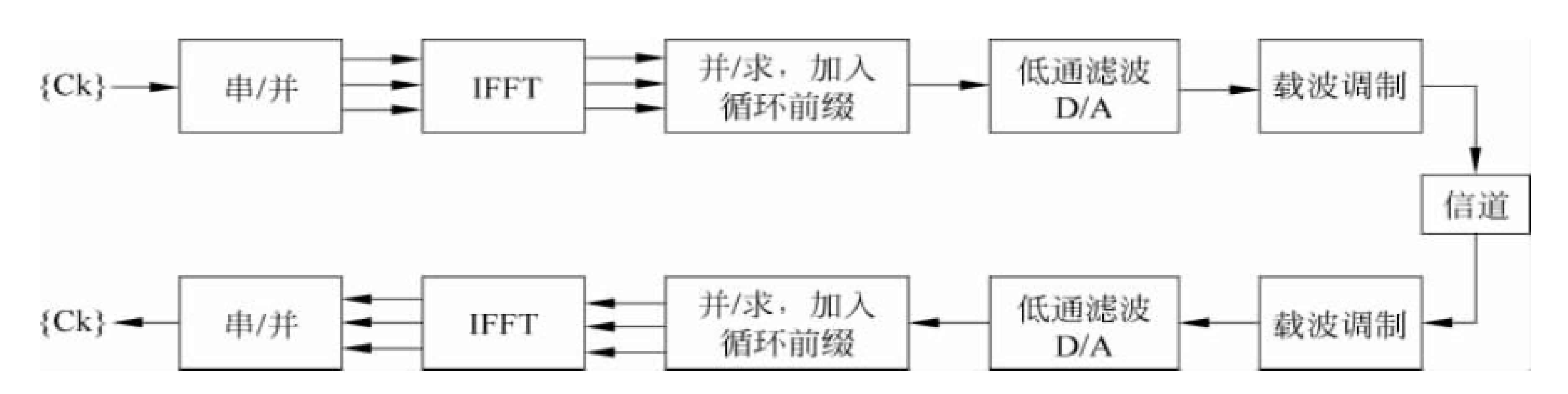


图 2-19　OFDM 系统结构框图

OFDM 系统的基本原理就是将指配的信道分成许多正交子信道，在每个子信道上进行窄带调制和传输，信号带宽小于信道的相关带宽。OFDM 发射/接收机的原理框图如图 2-20 所示。OFDM 信号的发送过程需要经过下面几个步骤：

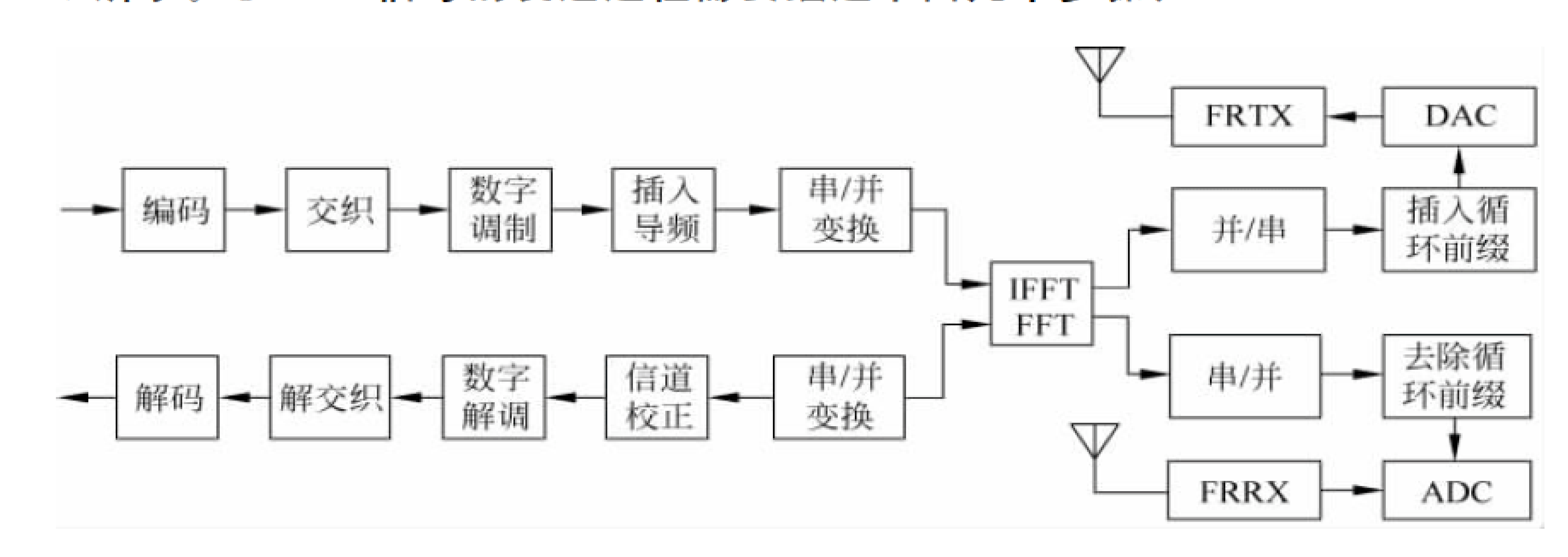


图 2-20　OFDM 发射/接收机的原理框图

(1) 编码。在基于 OFDM 调制技术的系统中，编码采用 Reed-Solomon 码、卷积纠错码、维特比码或 TURBO 码。

(2) 交织。交织器用于降低在数据信道中的突发错误，交织后的数据通过一个串并行转换器，将 IQ 映射到一个相应的星座图上。在这里，I 代表同相信号，Q 代表正交信号。

(3) 数字调制。在 OFDM 方式中，采用星座图将符号映射到相应的星座点上。这一过程产生 IQ 值，它们被过滤并进行 IFFT 变换。

(4) 插入导频。为了能够使接收稳定，在每 48 个子载波中插入 4 个导频信息。

(5) 串并转换。使串行输入的信号以并行的方式输出到 M 条线路上。这 M 条线路上的任何一条上的数据传输速率则为 R/M 码字/秒。

(6) 快速傅里叶逆变换。快速傅里叶逆变换可以把频域离散的数据转化为时域离散的数据。由此，用户的原始输入数据就被 OFDM 按照频域数据进行了处理。

(7) 并串转换。用于将并行数据转换为串行数据。

(8) 插入循环前缀并加窗。循环前缀为单个的 OFDM 符号创建一个保护带，在信噪比边缘损耗中被丢掉，以极大地减少符号间干扰。

接收器完成与发送器相反的操作。接收器收到的信号是时域信号。由于无线信道

的影响发生了一定的变化，首先要通过训练序列定时和频率偏移进行估计，同时将符号的定时信息传送到去循环前缀功能模块，在这里训练序列和导频信息主要是用来信道纠错。然后将信号经过一个串行一并行的转换器，并且把循环前缀清除掉。清除循环前缀并没有删掉任何信息，循环前缀中的信息是冗余的，使用循环前缀是为了保证前文提到的卷积特性的成立。总体来说，整个接收过程需要经过下面几个步骤：①定时和频率同步；②去循环前缀；③串并转换；④快速傅里叶变换；⑤并串转换；⑥信道校正；⑦数字解调；⑧去交织；⑨解调。

2. OFDM 的关键技术

1）同步技术

在 OFDM 系统中，N 个符号的并行传输会使符号的延续时间更长，因此它对时间的偏差不敏感。对于无线通信来说，无线信道存在时变性，在传输中存在的频率偏移会使 OFDM 系统子载波之间的正交性遭到破坏，相位噪声对系统也有很大的损害。

由于发送端和接收端之间的采样时钟有偏差，每个信号样本都一定程度地偏离它正确的采样时间，此偏差随样本数量的增加而线性增大，尽管时间偏差破坏子载波之间的正交性，但是通常情况下可以忽略不计。当采样错误可以被校正时，就可以用内插滤波器来控制正确的时间进行采样。

相位噪声有两个基本的影响，其一是对所有的子载波引入了一个随机相位变量，跟踪技术和差分检测可以用来降低共同相位误差的影响；其二因为相位误差导致子载波的间隔不再是精确的 $1/T$，也会引入一定量的信道间干扰(ICI)。

载波频率的偏移会使子信道之间产生干扰。OFDM 系统的输出信号是多个相互覆盖的子信道的叠加，它们之间的正交性有严格的要求。无线信道时变性的一种具体体现就是多普勒频移，多普勒频移与载波频率以及移动台的移动速度都成正比。多普勒展宽会导致频率发生弥散，引起信号发生畸变。从频域上看，信号失真会随发送信道的多普勒扩展的增加而加剧。因此对于要求子载波严格同步的 OFDM 系统来说，载波的频率偏移所带来的影响会更加严重，如果不采取措施对这种信道间干扰(ICI)加以克服，系统的性能很难得到改善。

OFDM 中的同步通常包括三方面的内容：帧检测、载波频率偏差及校正、采样偏差及校正。

由于同步是 OFDM 技术中的一个难点，因此，很多人也提出了很多 OFDM 同步算法，主要是针对循环扩展和特殊的训练序列以及导频信号来进行，其中较常用的有利用奇异值分解的 ESPRIT 同步算法和 ML 估计算法。对 OFDM 技术的同步算法的研究有很多，需要根据具体的系统具体设计和研究，利用各种算法融合进行联合估计才是可行的。OFDM 系统对定时频偏的要求是小于 OFDM 符号间隔的 4%，对频率偏移的要求大约要小于子载波间隔的 1%～2%，系统产生的 3dB 相位噪声带宽大约为子载波间隔的 0.01%～0.1%。

2）峰平比(PARP)的抑制

OFDM 包络的不恒定性可以用 PAPR 来表示。PAPR(peak-to-average power ratio)是峰值功率与平均功率之比。PAPR 越大，系统的包络的不恒定性越大。因此要

改善系统性能，就是要设法减小 PAPR。

由于 OFDM 信号为多个正弦波的叠加，当子载波个数多到一定程度时，由中心极限定理，OFDM 符号波形将是一个高斯随机过程，其包络是不恒定的。这种现象在非线性限带信道中是不希望出现的，经非线性放大器后，包络中的起伏虽然可以减弱或消除，但与此同时却使信号频谱扩展，其旁瓣将会干扰临近频道的信号。这在 OFDM 系统中将引起相邻信道之间的干扰，破坏其正交性。一般而言，发射机中的高频放大器 HPA 具有很强的非线性特征。因此如果不能改善 OFDM 对非线性的敏感性，OFDM 技术将不能用于使用电池的传输系统，如手机等移动设备。一般通过以下几种技术解决：

（1）限幅（clipping）技术。限幅技术是一种简单而有效的降低 PAPR 的方法，但是它会导致带内信号的失真和带外频谱弥散，从而使误码率性能恶化。

（2）编码技术。分组编码的方法既可以绝对地降低 PAPR，也具有一定的纠错能力。OFDM 信号的复包络依赖于发送数据信号序列的非周期自相关函数旁瓣。如果旁瓣小，则信号的起伏就小，即 PAPR 小，就可以得到准恒定（quasi-constant）幅度信号。因此，需要寻找自相关函数旁瓣小的发送信号序列。Golay 二进制序列（即 complementary）就是一种旁瓣小的序列。即使是它扩展到多相位序列，也仍然满足旁瓣小的特性。

（3）扰码技术。采用扰码技术，使生成的 OFDM 的互相关性尽量为 0，从而使 OFDM 的 PAPR 减少。这里的扰码技术可以对生成的 OFDM 信号的相位进行重置，典型的有 PTS 和 SLM 两种。

3）训练序列和导频及信道估计技术

接收端使用差分检测时不需要信道估计，但仍需要一些导频信号提供初始的相位参考，差分检测可以降低系统的复杂度和导频的数量，但却损失了信噪比。尤其是在 OFDM 系统中，系统对频偏比较敏感，所以一般使用相干检测。

在系统采用相干检测时，信道估计是必须的。此时可以使用训练序列和导频作为辅助信息，训练序列通常用在非时变信道中，而在时变信道中一般使用导频信号。在 OFDM 系统中，导频信号是时频二维的。为了提高估计的精度，可以插入连续导频和分散导频，导频的插入需要满足二维 Nyquist 采样定理，导频信号之间的间隔取决于信道的相干时间和相干带宽，在时域上，导频的间隔应小于相干时间；在频域上，导频的间隔应小于相干带宽。在实际应用中，导频模式的设计要根据具体情况而定。

OFDM 适用于多业务、高灵活性的通信系统，频谱利用率高，系统稳定性好，因此在很多领域得到了广泛的应用。欧洲的 DAB 系统使用的就是 OFDM 调制技术，使得在传输过程中能够实现低延迟、高速率的数据传输。54Mb/s 的带宽也基本上能够满足大部分用户对无线网络的要求。随着 OFDM 技术的不断完善，它的应用范围将会扩展到各个领域。

2.4.3 分集接收技术

无线信道是随机时变信道，在陆地通信系统中存在着多径干扰和衰落，在城市环境中衰落尤为严重。当不同的多径分量其衰落相互独立时，常采用分集接收、Rake 接收和均衡技术对抗衰落。在第三代移动通信中，分集接收技术得到了更加广泛的应用。

1．分集接收技术的基本原理

与均衡不同，分集接收技术不需要训练码，从而节省了开销。分集接收技术通过查找和利用自然界无线传播环境中独立或者至少是高度不相关的多径信号来实现。在实际应用中，分集的各个方面的参数都由接收机决定，发射机并不知晓分集的情况。

分集的概念可以简单地解释为：如果一条无线传播路径中的信号经历了深度衰落，而另外一条相对独立的路径中可能仍包含着较强的信号，从而可以在多径信号中选择两个或者两个以上的信号。其基本原理是：在多径环境中，如果某一径的强度低于检测门限值的概率为 p，则在 L 径情况下，所有 L 个径的强度都低于检测门限的概率为 P_1，远低于 p。分集接收技术的代价是增加了接收的复杂度，分集的好处是它对于接收端的瞬时信噪比和平均信噪比都有提高，通常可以提高 20～30dB。

为了在接收端得到几乎相互独立的不同路径，可以通过空域、时域、频域的不同角度、不同的方法与措施加以实现，其中最基本的有如下几种。

1）空间分集

空间分集，又被称为天线分集，是无线通信中使用最多的分集形式，它既可以用于移动台，也可以用于基站，或者同时应用于两者。空间分集基于这样一个事实，在移动台端，如果天线间的相隔距离等于或者大于半个波长，或者在基站端分集天线间的相隔距离大于一定值（通常是几十个波长），那么不同的分集天线上收到的信号包络将基本上是不相关的。

空间分集还有以下两类变化形式：极化分集和角度分集。

（1）极化分集。它是利用在同一地点两个极化方向相互正交的天线发出的信号可以呈现不相关的衰落特性来进行分集的，其优点为结构紧凑、节省空间；但缺点是由于发射功率要分配到两副天线上，因此有 3dB 的损失。

（2）角度分集。由于地形、地貌、接收环境的不同，使得到达接收端的不同路径的信号可能来自不同的方向，这样在接收端可以采用方向性天线，分别指向不同的到达方向。而每个方向性天线接收到的多径信号是不相关的。

2）频率分集

频率分集是在多个载频上传送信号，它基于在信道相干带宽之外的频率上不会出现相同的衰落的原理，将待发送的信息分别调制到不同的载波上发送至信道。在频率分集中，不同的载波之间的间隔 Δf 需足够大且大于频率相干带宽 ΔF。

频率分集与空间分集相比，其优点是减少了接收天线与相应设备的数目；缺点是占用更多的频谱资源，而且有可能需要和频率分集中采用的频道数相等的接收机，所以一般应用在特殊业务中。

3）时间分集

时间分集是指以超过信道相干时间的时间间隔重复发送信号，以便让再次收到的信号具有独立的衰落环境，从而产生分集效果。对于一个随机衰落信号，如果取样时间间隔 Δt 足够大且大于相干时间 ΔT 时，才可以构成时间分集。

在时间分集中，将待发送的信号每隔一定时间间隔重复发送，在接收端就可以得到

N 条独立的分集支路。

时间分集与空间分集相比，其优点是减少了接收天线的数目，缺点是要占用更多的时隙资源，从而降低了传输效率。

分集技术的优势是很强大的，在很多通信系统中都采用了分集技术。而且在这里需要强调的是，在一个通信系统中采用一种以上的分集方式并不是多此一举。例如在 IS-95 系统中，同时采用了空间分集、频率分集、时间分集技术，它们的目的都在于以最小的发射概率得到所需要的误码率。

分集接收中，在接收端从 N 个不同的独立信号支路所获得的信号可以通过不同形式的合并技术来获得分集增益。合并时采用的准则和方式主要可以分为三种：最大比值合并、等增益合并、选择式合并等。

综合三种合并方法，最大比值合并的性能最好，等增益合并实现较为简单，性能次之，选择式合并将未被选择的路径弃之不用，性能最差。

2．Rake 接收机

Rake 接收机的基本原理是将那些幅度明显大于噪声背景的多径分量取出，对它进行延时和相位校正，使之在某一时刻对齐，并按一定的规则进行合并，变矢量合并为代数求和，有效地利用多径分量，提高多径分集的效果。

由于用户的随机移动性，接收到的多径分量的数量、幅度大小、时延、相位均为随机量，因此合成后的矢量亦为一个随机变量。

若无 Rake 接收机，多径信号的合成如图 2-21(a)所示，而采用 Rake 接收机后，多径信号的合成可改变成图 2-21(b)所示。



图 2-21　多径信号的矢量合成图

可见，通过 Rake 接收，将各路径加以分离，并进行相位校准后加以利用，变矢量相加为代数相加，有效地利用了多径分量。

根据 CDMA 系统中可分离的径的概念，当两信号的多径时延相差大于一个扩频码片宽度时，可以认为这两个信号是不相关的，或者说是路径可分离的。反映在频域上，即信号的传输带宽大于信号的相干带宽时，认为这两个信号是不相关的，或者说是路径可分离的。由于 CDMA 系统是宽带传输的，所有信道共享频率资源，所以 CDMA 系统可以使用 Rake 接收技术，而其他两种多址技术 TDMA、FDMA 则无法使用。

Rake 接收机分集的度量取决于多径时延宽度和多径分离的能力。Rake 接收机的框图如图 2-22 所示。

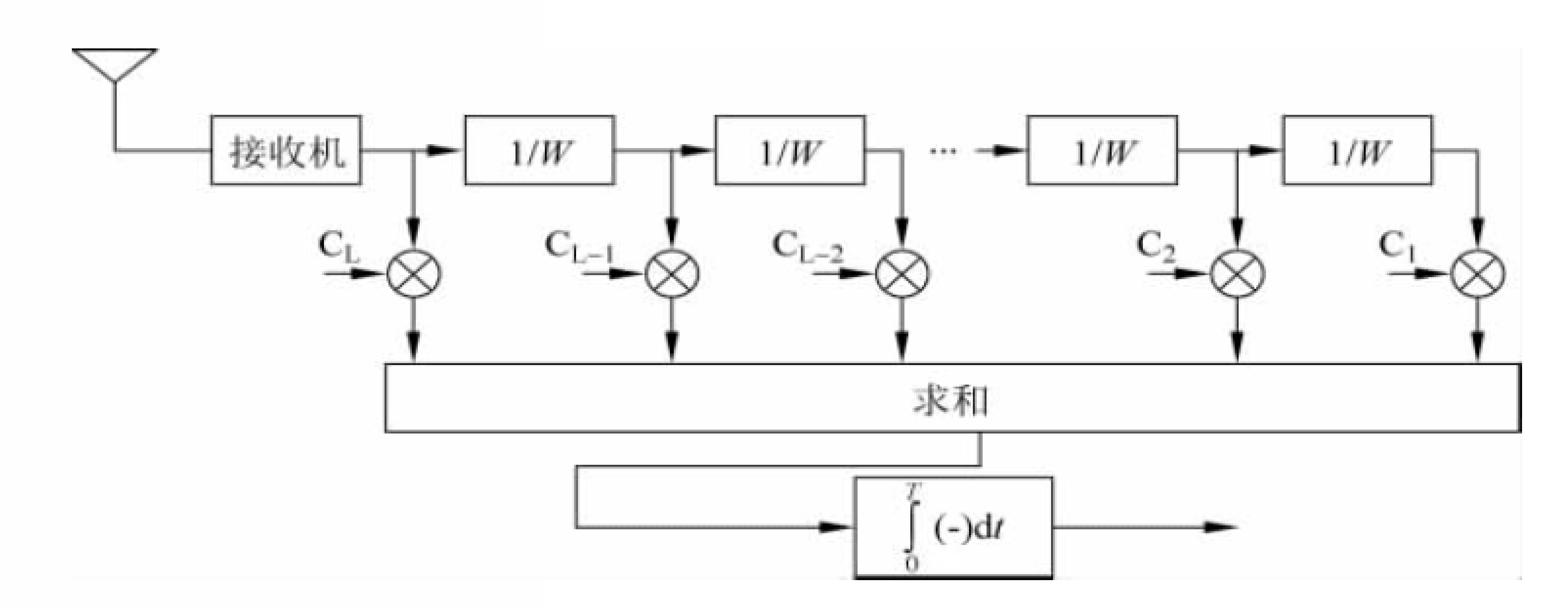


图 2-22　Rake 接收机框图

2.4.4　自适应编码调制技术

自适应编码调制技术是智能技术在通信领域的应用。由于移动通信信道特性复杂，不同频率、不同时间、不同地域信道的特性都不一样，而移动通信中有多种编码与调制的方案，不同的编码与调制的方案抗噪声性能不一样，实现的复杂度也不一样。如果能根据移动通信信道的具体特性选择恰当的编码与调制的方案，则可以节约通信资源，使得用户获得更高的信息速率。由于不是调整发射功率，而是调整编码与调制方式来进行信道适应，所以系统中的干扰变化不再那么剧烈。

无线信道的一个很重要的特点就是具有很强的时变性，短时间瑞利衰落可以达到十几甚至几十 dB。对这种时变特性进行自适应跟踪会给系统性能的改善带来很大的好处。链路自适应技术就是根据时变衰落信道的变化，通过自适应调整发射功率、符号速率、调制阶数、编码速率、编码方案或上述几个因素的组合来实现链路预算的实时平衡，达到增加系统容量和改善通信质量的目的。

AMC(adaptive modulation and coding)技术就是一种自适应技术，它能根据信道状态自适应调节调制方式、信道编码方式和码率，从而提高了频带利用率，使用户达到尽量高的数据吞吐率。

自适应技术通常包含三个部分：对变化情况的测量、估计或预测；所需改变参数的选择；给其他部分的信令信息。

要适应变化，首先要了解变化的情况，这是自适应技术的前提，因此要对时变信道做出质量估计。信道估计需要确定合适的物理量，通常使用的指标有信噪比(SNR)、载噪比(CNR)、均方误差(MSE)、比特误码率(BER)等，这些值可以在接收端或发送端进行测量，此外还要考虑自适应速度和估计算法的复杂度等。

在自适应调制中可以改变的参数有调制方式和发送功率等。最佳参数的选择就是在系统限定的条件下，目标函数的最优化问题。例如，在速率和发送功率固定的条件下，使误码率最小；或者保证一定的误码率和发送功率，使传送速率最大等等。而在实际中可以有一些简单的方法，如设定相应的变化门限等[12]。

AMC 主要有以下两个优点：

(1) 处于有利位置的用户可以得到更高的数据速率，提高小区平均吞吐量。

(2) 链路自适应地改变调制编码方案代替改变发射功率，减小了冲突。

而实现 AMC 还面临一些挑战：

(1) 为了选择调制和编码方式，基站必须知道信道质量，AMC 对信道测量误差和延时敏感。

(2) 信道质量估计的错误会使基站选择不合适的数据速率。

(3) 由于移动信道是不断变化的，信道测量报告的时延会使得信道质量报告不太可靠。

自适应调制技术是自适应无线传输的一种，也是未来无线通信的一种关键技术。自适应调制可以通过改变调制方式来适合不同系统的传输质量需求，从而优化系统的性能。分析和仿真表明，根据信道的优劣情况进行自适应调制的系统在保持较低的 BER 性能的同时，尽可能提高了系统的性能，从而使传输系统得到优化。

2.5 组网技术的融合

随着移动通信系统和移动网络的发展，未来的移动通信将是全球一体化的无线通信系统，虽然 OFDM、MIMO、智能天线等技术的提出解决了高速传输和频谱效率的矛盾，但高速传输与覆盖之间的矛盾仍是需要解决的问题之一，于是在此情势下提出了多跳中继技术。在多跳模式下，网络可以为每个用户提供多条传输路径。自组织网络（Ad Hoc Network）、无线传感器网络（WSN）均属于移动多跳网络的范畴。而协作通信则从多个节点乃至整个网络的角度考虑系统的联合优化，进一步推动了无线通信的发展。在协作通信系统中，多个节点通过协作和共享资源能够有效抵抗无线信道的衰落、充分地挖掘网络的通信性能。

2.5.1 多跳中继技术

作为下一代移动通信系统的候选关键技术之一，无线多跳中继技术就是在基站与移动台之间增加一个或多个中继节点，对无线信号进行一次或者多次的转发，即无线信号要经过多跳才能到达移动台，能以有效的方式提供覆盖扩展及吞吐量和容量的增强，并由于中继节点放置的高度灵活性，使得网络快速部署，从而适配用户吞吐量的变化。

1. 中继网络模型及应用场景

经典的中继网络模型可以看作由一个信源（MR-BS）、一个中继站（RS）和一个信宿（MS）组成，其中信源即多跳基站。信宿接收信号有两个路径：一个是单跳直接路径，另一个是两跳中继路径，即从 MR-BS 到 RS 和从 RS 到 MS。其中，MR-BS 同 RS 之间的物理信道称为中继链路，而 RS 到 MS 和 MR-BS 同 MS 之间的物理信道称为接入链路。该模型也可当作大型中继系统的基本单元，如图 2-23 所示。

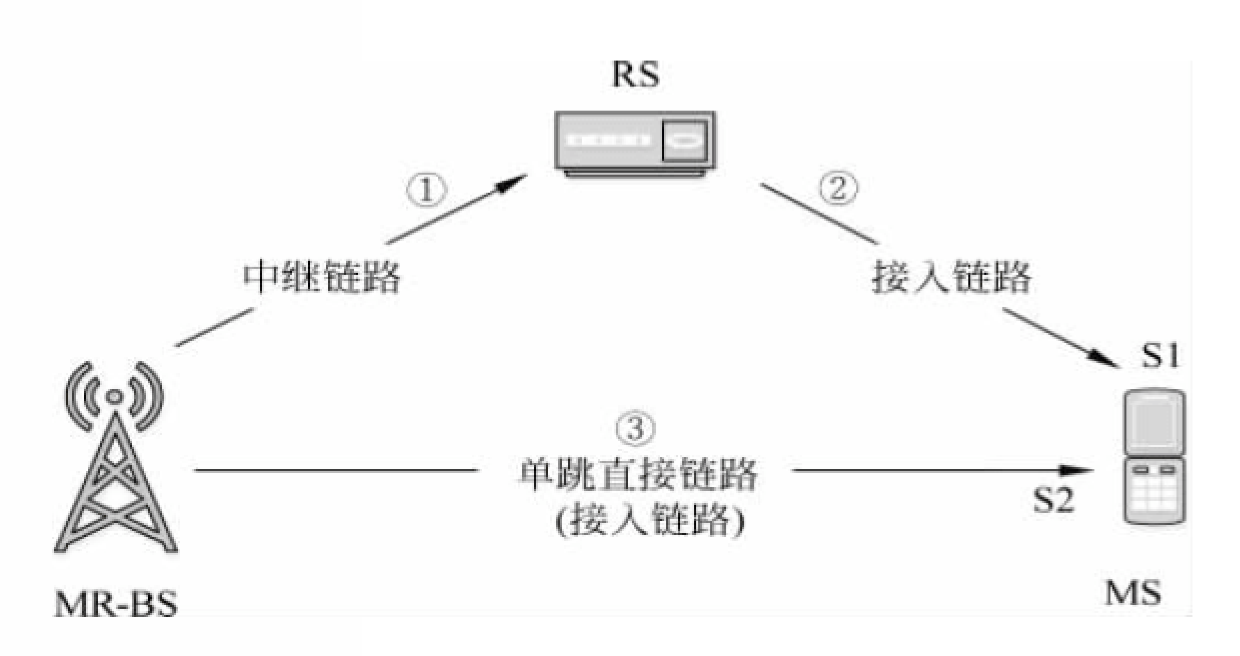


图 2-23　三节点的中继网络模型

将以上中继网络模型抽象为如图 2-24 所示[13]。

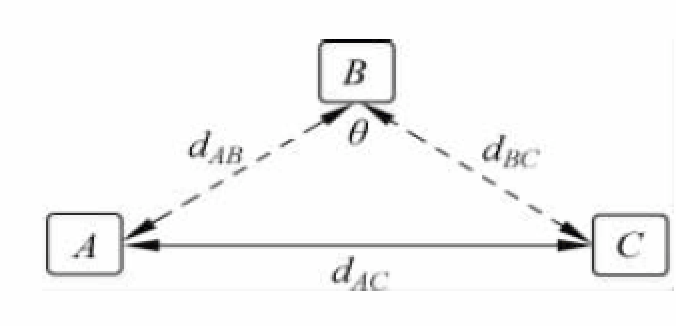


图 2-24　中继模型抽象图

其中，A 为信源，B 为中继节点，C 为信宿，设单跳路径距离为 d_{AC}，两跳中继路径的距离分别为 d_{AB} 和 d_{BC}，假设路径损耗为 2，B 和 C 接收的功率为 P，则单跳发送功率 P_{direct} 和两跳发送功率 P_{relay} 可分别表示为

$$P_{\text{direct}} = P(d_{AB}^2 + d_{BC}^2 - 2d_{AB}d_{BC}\cos\theta) \qquad (2\text{-}4)$$

$$P_{\text{relay}} = P(d_{AB}^2 + d_{BC}^2) \qquad (2\text{-}5)$$

由以上公式可以看出，若 θ 为钝角，则 P_{relay} 小于 P_{direct}。而当中继节点位于信源和信宿中点时，多跳发射功率达到最小。

中继站通常情况下都工作于半双工模式下，即中继站使用不同的信道进行接收和发送，且用于中继操作的信道一般会由两个正交的子信道组成。多跳中继的应用场景主要有以下三种，如图 2-25 所示。

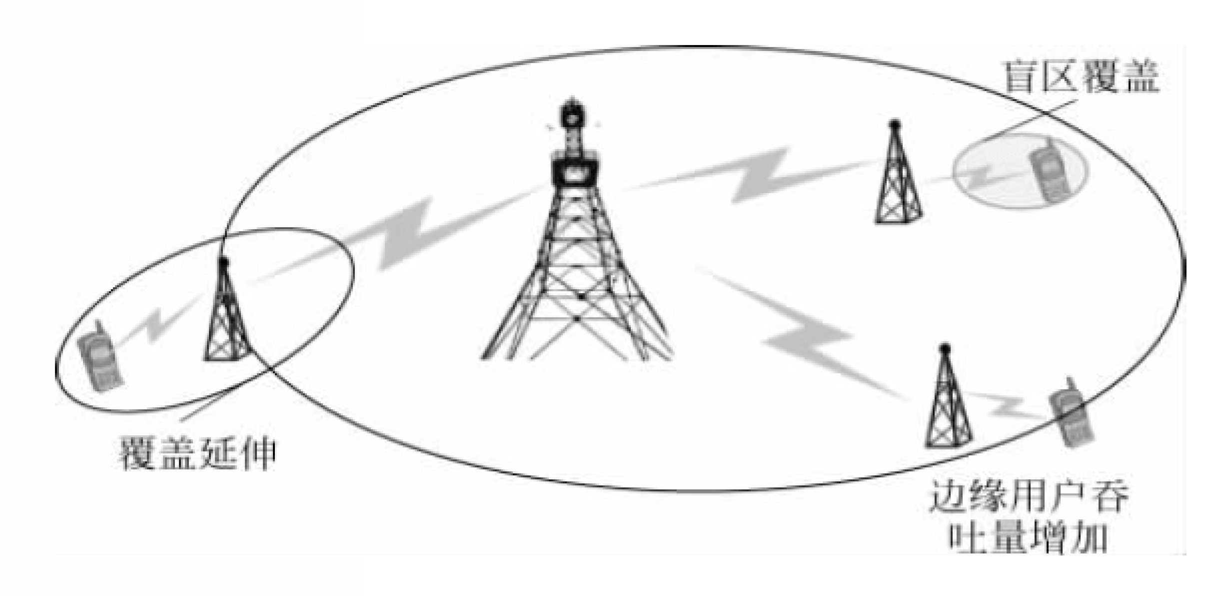


图 2-25　多跳中继应用场景

(1) 改善小区边缘性能。

(2) 小区覆盖区域内的覆盖盲区。

(3) 提供常规小区范围以外的覆盖。

2. 中继转发策略

中继节点作为在数据接收和转发过程中有数据中转功能的节点，基于其如何处理接收信号，可分为以下几种方式：放大转发模式(amplifying and forwarding，AF)、译码转

发模式(decoding and forwarding,DF)、估计转发模式(estimating and forwarding,EF)、编码协作模式(coded cooperation,CC)。

AF 模式是一种最简单的中继转发模式,它和译码转发模式并称为非再生中继,中继的复杂度要求较低。中继节点不对接收的信号进行解调和解码,而是直接将收到的信号进行模拟处理后转发,这是采用最早的一种协作模式。但中继接收到的信号不仅包含有用信号,还包含混入的噪声。在信号放大的过程中,有用信号和噪声被一起放大和转发,因而会提高系统噪声水平,引入噪声放大效应。目的节点经合并处理后提取出有用信号。

相较于单跳传输链路,该转发模式在目的节点处还可以获得较好的判决效果,但是需要在信噪比较高的环境下才易于实现。

DF 模式,又称为再生中继。在该模式中,中继对接收到的信号进行解调、译码(视是否编码而定)和估计。然后再进行编码和调制后发送出去。该模式相较于放大转发模式有更好的性能,在发送之前,对消除噪声的有用信号重新进行编码调制有效地消除了信号中的噪声,避免信号的恶化[14]。

但 DF 模式受信源—中继端信道传输性能影响较大,若编码方式不采用 CRC(cyclical redundancy Check)码,得不到满分集阶数。中继节点对源节点信息解码错误所带来的误差会随着跳数的增加而不断累积,从而影响到分集效果和中继性能。这表明源—中继节点信道传输特性的好坏对 DF 方式协同通信系统的性能有很大影响。

EF 模式中,RS 不对信号进行译码,而是将接收到的信号进行量化压缩后产生一个估计值,再将这个估计值转发出去。转发后的信号中不仅包含正确的信号,还可能会包含估计的错误信号,目的节点会利用 RS 的估计作为边信息进行检测。这种模式是在"解码转发"模式不可用的情况下提出的,它吸收了前两种模式的优点。

编码协作模式(CC)对正确解出的合作伙伴的信号重新进行编码。即中继节点接收来自于基站端已被噪声干扰的信号,同时,它将对信号进行解码处理,以获得原始信息。然后再将其获得的信息重新编码,以一定功率发送给用户端。

在慢衰落条件下,即使移动终端间的信道传输特性非常差,编码协作依然能够显著提高这两个移动终端的误码率性能;若协作双方都能够正确解码,则系统可以获得满分集增益;而在快衰落条件下,编码协作会牺牲上行信道传输特性相对较好的终端的性能。

3. 中继部署方案

按照中继部署时的目的不同,移动网络的中继部署可以分为以下三个方案[15]。

1) 增强用户数据速率的部署

该种部署中的中继是为了使处于蜂窝小区边缘的用户或者处于建筑阴影区域内的用户获得更高的信噪比(SINR),从系统角度看,该类中继可以辅助基站为其覆盖范围内的用户提供公平的服务调度以及均衡的传输速率。

如图 2-26 所示为该类中继部署场景中的一个小区的基本架构。该小区中配置了多个中继站,其中处于边缘区域的中继如 RS1、RS2、RS3 和 RS4 为用户提供高速的接入服务,其基站和用户间的跳数最大为两跳,而 RS5 和 RS6 则是为处于阴影效应区域中的用户提供高速的中继转发服务,跳数可能大于两跳。

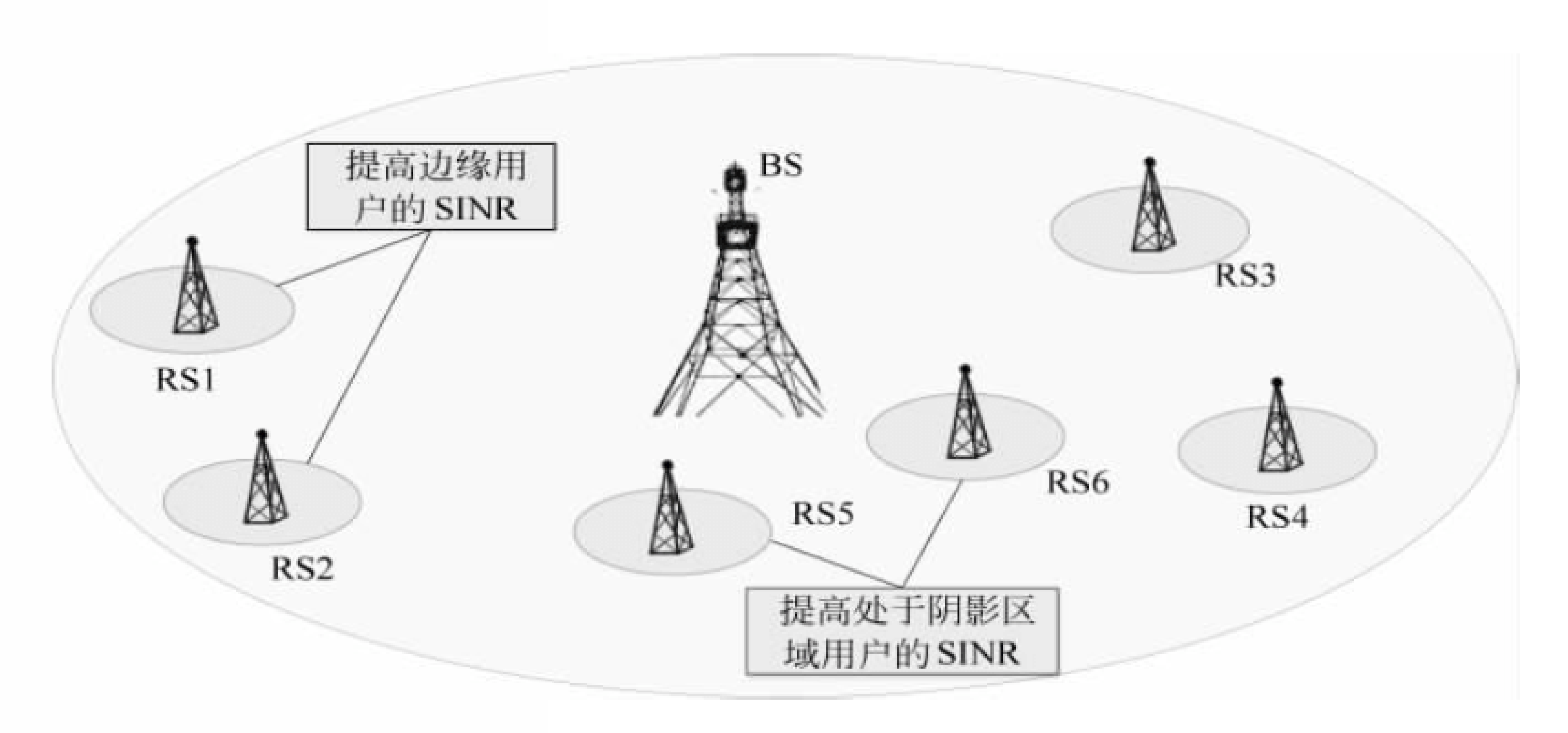


图 2-26　增强用户数据速率的中继

2）增加网络覆盖范围的部署

该类场景中的中继为了给处于小区外的用户提供服务，这些区域可能是不易部署基站的地方或者城市边缘没有覆盖的地方。

图 2-27 为扩大覆盖范围的移动小区中继部署架构图。基站位于该小区的中心，中继站位于传统小区覆盖范围之外，该场景中的中继拓扑一般是两跳以上的。

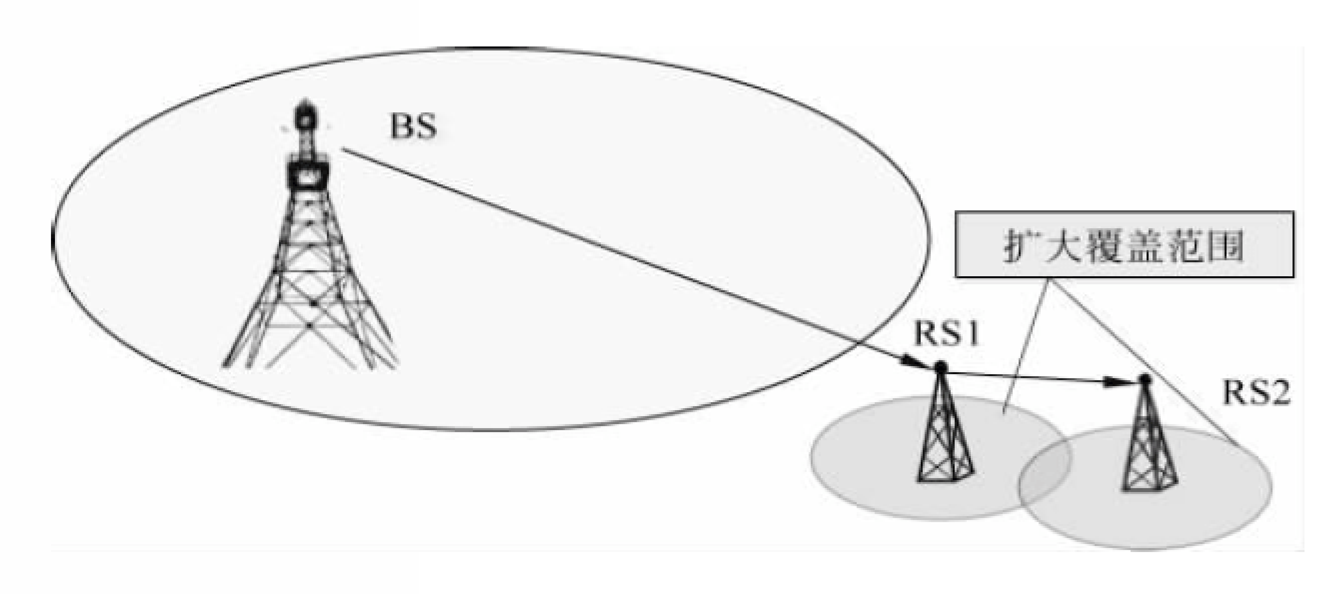


图 2-27　扩大小区覆盖范围的中继

3）增加网络容量的部署

该场景中的中继部署主要是为了通过允许频率复用以提供较高的系统容量，通过一定的调度方式增加高负载区的容量。图 2-28 为该场景下一个可能的架构图。其中基站和中继站都是正六边形覆盖，覆盖范围是相同的。

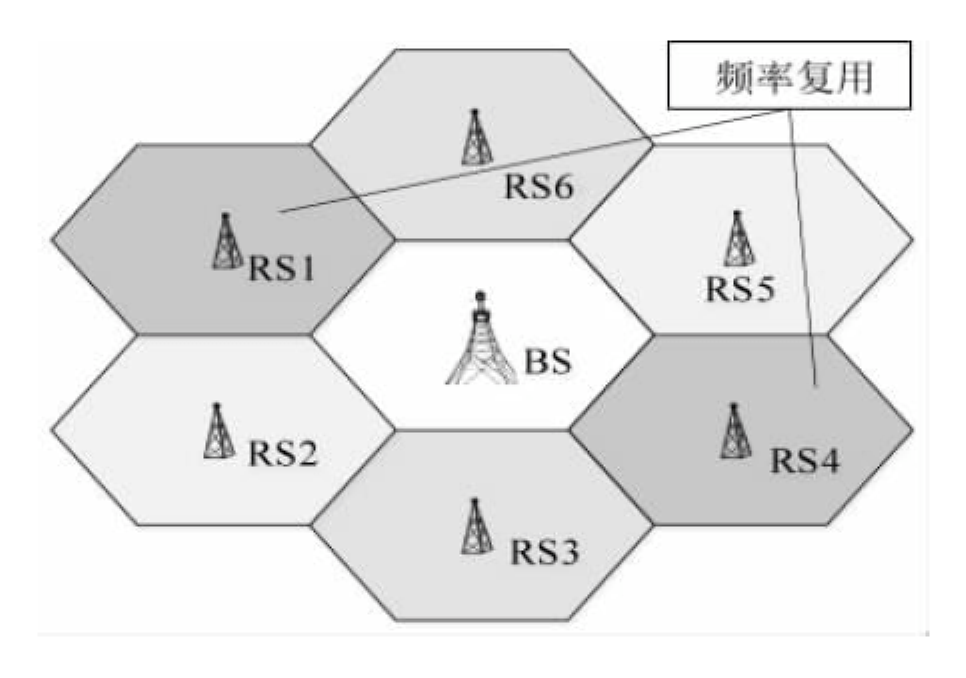


图 2-28　增加网络容量的中继

这种场景下可以实现频率重利用，如相隔较远的 RS1 和 RS4、RS2 和 RS5、RS3 和 RS6 可以通过基站一定的调度和功率控制方式在接入链路上使用相同的频率，增加小区的吞吐量。也可以利用不同的小区级间的跨小区协作调度来控制小区边缘的干扰。

WiMAX 中的中继部署就是以上三个场景的典型应用，如 IEEE 802.16j 标准，它是 IEEE 802.16e 的修订和补充版本，其目的就是研究如何在 WiMAX 中采用多跳中继技术，以增加网络的容量和覆盖范围，并改善小区的边缘效应、信号盲点等问题，降低组网成本。

IEEE 802.16j 的具体工作场景有以下几种[16]：

（1）固定建筑上的中继。这类中继在固定的区域为移动用户提供服务，一般建设在建筑物的顶端，用于增加小区内的吞吐量和覆盖范围。

（2）建筑物内部的中继。在很多建筑物的内部，或隧道、地铁等区域中，由于阴影效应的存在，导致通信质量和数据业务质量难以保证用户需求，建筑物内的中继可以提升通信质量，克服室内阴影效应等。

（3）临时中继。临时中继可以将人群密集地方的通信需求分流到附近小区中去，还可以在原有基站设备遭到破坏的情况下使用。

（4）在高速移动的交通工具上的中继。随着高铁的普遍使用，在高铁等速度较快、人群密集的交通工具上的通信需求成为挑战，移动车辆上的中继给用户提供了可靠的通信覆盖，方便了人们生活。

2.5.2 自组织网络

1. 自组织网络简介

传统的无线移动网络，需要固定的网络基础设施的支持。而自组织网络（Ad Hoc）没有固定的设备支持，网络中的各节点可以随意移动并自行组网，通信时，由其他用户节点进行数据的转发。

在自组织网络中，每一个节点都既能作为主机也可作为路由器使用。作为主机，终端需要运行用户所需要的应用程序；而作为路由器，终端需要运行相应的路由协议，根据遵循的路由策略进行分组转发和路由维护等工作。由于终端的无线传输范围有限，两个无法直接通信的终端节点往往会通过多个中间节点的转发来实现通信，即要进行多跳。

自组织网络突破了传统无线蜂窝网络的地理局限性，能够更加快捷、高效地部署网络，适合于一些紧急场合的通信需要，如事故的突发现场以及人们希望能迅速共享信息的办公室、会议等场所。但无线自组织网络也存在网络带宽受限、对实时性业务支持较差、安全性不高等缺点。

由于自组织网络是一种移动、多跳、自律式系统，因此它具有以下几个显著特征：

（1）无中心化节点的分布式特征。自组织网络是一个对等性网络，网络中所有节点的地位平等，无须设置任何的中心控制节点，网络的建立和调整是通过各个节点的有机配合实现的，自组织网络均衡了各个节点的特殊性，各节点在控制能力上没有重要和次要之分，从而防止由于控制中心被破坏而引起全网瘫痪的危险，具有很强的抗毁性和健壮性。

（2）网络拓扑结构动态变化。由于用户终端的移动具有很大的随机性，加上发射功率的变化、信道间的相互干扰以及复杂地形等综合因素的影响，网络的拓扑可能会以难

以预测的方式随时发生变化。

(3) 自组织性。自组织网络不需要人工干预和任何其他基础设施的支持,其节点能够遵循一种自组织原则,自动探测网络的拓扑信息,快速检测其他节点的存在和探测其他节点的能力集,从而自动地选择传输路由。即使网络发生动态变化或某些节点严重受损时,仍可迅速调整其拓扑结构以保持必要的通信能力。

(4) 信道质量较差。自组织网络采用无线传输方式进行通信。但由于物理信道本身的衰减大、干扰大、多径效应等特点,使其信道质量比有线信道差的多。并且由于多个节点分布式竞争使用信道,使得每个节点实际使用的带宽远小于物理层提供的最大传输速率。

(5) 设备动力受限。移动 Ad Hoc 网络跨越多个不同的终端,由于网络节点的移动特征,其中大多数节点以电池作为动力,另外,设备处理能力也会影响设备本身特性。因而,在电池容量没有大幅提高前,节省功率将是移动 Ad Hoc 网络技术中一个需要高度重视的问题。

(6) 安全性面临挑战。自组织网络由于采用无线信道、有线电源、分布式控制等技术,更容易受到被动窃听、主动入侵、拒绝服务、数据篡改和重发、伪造身份、剥夺"睡眠"等各种方式的攻击。但由于终端电源的有限性,自组织网络无法实现复杂的加密算法,增加了被窃密的可能性。从而信道加密、抗干扰、用户认证等安全措施都需要进行考虑。

(7) 单向的无线信道。由于移动 Ad Hoc 网络采用无线信道作为通信媒介,受地形环境、发送功率和接收灵敏度等因素影响,无线信道可能是单向信道。

2. 自组织网络研究的关键技术

移动 Ad Hoc 网络作为一种新型的网络通信支撑环境,其独特的网络特征使得其路由问题、安全问题、QoS 等问题成为这一领域的研究热点与难点。目前,自组织网络研究的主要问题包括以下几个方面:

(1) 物理层自适应技术。由于能量的限制,自组织网络的链路层设计面临许多新的挑战。由于多径衰落引起的幅度与相位的扰动、延迟扩展引起的码间串扰、来自其他节点信号的干扰等因素,使得无线信道的单位带宽容量相对较小。如何充分利用有限的带宽、能量资源,基于应用的特点和对 QoS 的要求,最大化网络的吞吐量,最小化能量的消耗,延长能量受限网络的寿命,将是链路自适应技术要解决的问题。解决的方法主要采用自适应编码、自适应调制、帧长自适应、自适应功率控制、自适应资源分配、自适应链路调整等技术。

(2) 信道接入技术。信道接入技术控制着节点如何接入无线信道,对 Ad Hoc 网络性能起着决定性的作用。Ad Hoc 网络的无线信道不同于普通网络的共享广播信道、点对点无线信道和蜂窝移动通信系统中由基站控制的无线信道,它是多跳共享的多点信道。即一个节点发送信息时,邻居节点(也只有邻居节点)可以收到。此外,Ad Hoc 网络还存在独特的隐藏终端和暴露终端的问题[25]。

(3) 路由技术。路由协议是 Ad Hoc 网络的重要组成部分。要实现多跳路由,必须要有路由协议的支持。传统的路由协议并不支持 Ad Hoc 网络。必须为它设计专门的路由协议。Ad Hoc 网络路由协议面临非常艰巨的技术挑战。从无线信道带宽窄的角度

看,路由协议在节点间交互的信息应尽量少,以减小路由协议的开销,提高信道的效率。而从另一方面来看,与固定网络相比,由于节点的移动性,Ad Hoc 网络的拓扑变化要频繁得多,为了能够尽快、尽量精确地反映网络拓扑的变化,就需要更加频繁地在节点间交互控制报文。这本身就是一对矛盾。因此,设计一个在所有情况下都普遍适用的 Ad Hoc 网络路由协议基本上是不太可能的。已有的 Ad Hoc 网络路由协议从不同的角度对无线多跳路由问题进行了研究,对某一个或几个指标进行了优化,适应于不同的环境。IETF 成立的 MANET 工作组目前就主要负责 Ad Hoc 网络 IP 层路由的标准化工作。

(4) QoS 保证。Ad Hoc 网络出现的初期,主要用于传输少量的数据信息。随着应用的不断扩展,需要在 Ad Hoc 网络中传输语音、音像等多媒体信息。多媒体信息对带宽、时延、时延抖动等都提出了很高的要求。这就需要提过一定的服务质量保证。Ad Hoc 网络中的服务质量保证是个系统性问题,不同层都要提供相应的机制。例如,应用层要提供自适应信源码和压缩技术,网络层要提供 QoS 路由,链路层要提供资源预留策略等。

(5) 广播和多播。由于 Ad Hoc 网络的特殊性,广播和多播问题也变得非常复杂,它们需要链路层和网络层的支持。

(6) 安全问题。Ad Hoc 网络的缺点之一就是安全性较差,易受窃听和攻击。

(7) 网络管理。网络管理的范围较广,包括 Ad Hoc 网络中的服务发现、移动性管理、地址管理、服务管理等。要有相应的机制解决节点定位、地址自配置等问题。

(8) 传输层问题。在 Ad Hoc 网络中,要对传输层服务进行修改,以解决无线信道的衰落、干扰、节点移动等因素造成的报文冲突和丢失,满足数据传输的需要。

3. 自组织网络路由协议

由于网络层路由协议对自组织网络的重要性,使其成为研究的一个热点。到目前为止,已经有相当多的自组织网络路由标准推出。根据路由表建立方和维护特点的不同,将现有的路由协议分为三类:路由表驱动型路由协议、按需(on-demand)路由协议,以及两种模式的混合形式,如图 2-29 所示。

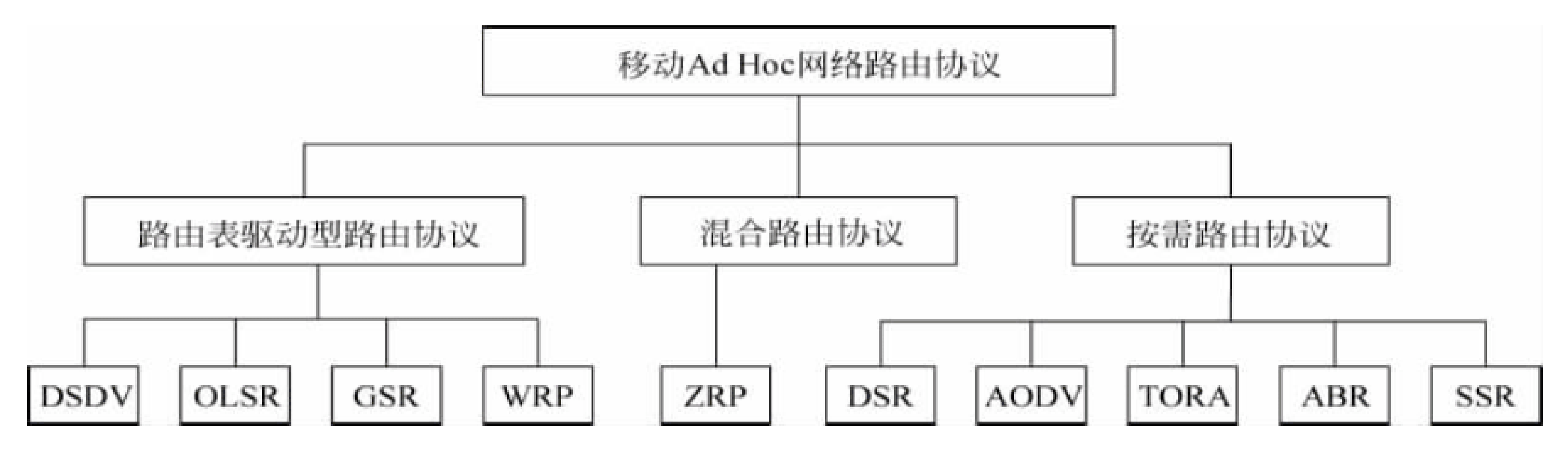


图 2-29 路由协议的分类

下面介绍几种典型的路由协议。

1) 路由表驱动型路由协议

路由表驱动型(table-driven)路由协议又被称为主动路由协议,是一种基于路由表的路由协议。在这种路由协议中,每个节点维护一张或多张路由表,包含到达网络中其他

所有节点的路由信息。当检测到网络拓扑结构发生变化时，节点在网络中发送更新消息。路由表驱动型路由协议不断地检测网络拓扑和链路质量的变化，并根据变化更新路由表，所以路由表可以准确地反映网络的拓扑结构。源节点一旦要发送报文，可以立即得到到达目的地的路由。典型的路由表驱动型路由协议包括 DSDV、OLSR、WRP 等协议。

（1）DSDV 路由协议。DSDV 协议是基于传统 Bellman-Ford 路由选择算法经改良而发展出来的，是一个基于路由表驱动的路由协议，它的最大优点是解决了传统距离矢量路由协议中的无穷环路问题。在 DSDV 路由协议中，每个节点都维护一张路由表，该路由表表项包括目的节点、跳数、下一跳节点和目的节点的序号。其中，目的节点的序号由目的节点分配，主要用于判断路由是否过时，并可防止路由环路的产生。每个节点必须周期性地与相邻节点交换路由信息，当然也可以根据路由表的改变来触发路由更新。路由表更新有两种方式：一种是全部更新，即拓扑更新消息中将包括整个路由表，主要应用于网络变化较快的情况；另一种方式是部分更新，更新消息中仅包含变化的路由部分，通常适用于网络变化较慢的情况。

但是在拓扑结构变化频繁的无线网络环境中，DSDV 可能存在一定的问题，一是节点维护准确路由信息的代价高，要频繁地交换拓扑更新信息；二是有的时候可能刚得到的路由信息随即又失效了。因此 DSDV 协议主要用于网络规模不是很大，网络拓扑变化相对不是很频繁的网络环境，而在拓扑变化频繁的网络中必须采用其他的方法。

（2）OLSR 路由协议。最优化链路状态（OLSR）协议是为移动 Ad Hoc 网络开发的。OLSR 协议是路由表格驱动、主动式路由协议，即有规律地与网络中其他节点交换拓扑信息。每个节点从相邻节点中选择一组节点作为“多点中继（multi-point relay，MPR）”。在 OLSR 协议中，只有被选作 MPR 的节点才负责转发控制消息，将控制消息传送到整个网络。

2）按需路由协议

按需路由协议也被称为反应式路由协议、源启动按需路由协议。通常，按需路由包括三个过程：路由发现过程、路由维护过程和路由拆除过程。与表驱动路由协议不同，按需驱动路由协议不需要周期性地进行路由更新。当源节点有路由需要时，它泛洪一个路由请求分组来构造一个路由。当收到路由请求分组后，目的节点基于某种路由选择算法来选择最好的路由。然后路由应答分组沿着这条选择的路由返回源节点。按需驱动路由的最大优点是它不需要周期性对路由信息进行广播，节省了移动 Ad Hoc 网络中的有限资源。它的缺点是等待路由发现将带来一定的时延圈。

下面将对一些重要的按需驱动路由协议进行分析和比较。典型的按需路由协议包括 DSR、AODV、TORA、ABR 等。

（1）DSR 路由协议。DSR（dynamic source routing）是一种基于源路由的按需驱动路由协议，它使用源路由算法而不是按逐跳路由的方法。网络中每一个节点需要维护一个已知路由表，并当发现新的路由时就更新该路由表。每一个数据包的包头都包含该数据包从源节点到目的节点路由所经过的中间节点序列信息，故称为源路由算法。

DSR 的优点在于 DSR 中，节点仅需要维护与之通信的节点的路由，协议开销较小；同时 DSR 使用路由缓存技术减少了路由发现的开销；此外，一次路由发现过程可能会产生多条到目的点的路由，这也将有助于 DSR 的路由选择。但是，由于每个数据包的头部

都需要携带路由信息，数据包的额外开销较大，路由请求消息采用泛洪方式，使得相邻节点路由请求消息可能发生传播冲突并可能会产生重复广播；由于缓存路由，过期路由会影响路由选择的准确性。

（2）AODV 路由协议。AODV 协议是一种按需路由协议，它根据业务需求建立和维护路由，是由 DSDV、DSR 路由协议算法发展而来的。在 AODV 协议中，当一个源节点希望建立通向某个目的节点的路径时，源节点发起一个路径发现过程，它广播路由请求分组（RREQ）给它的邻近节点（相互处于对方的无线电覆盖范围之内的节点），RREQ 再被这些邻近节点转发，直到到达目的节点或一个拥有通向目的节点足够新的路径的中间节点。中间节点收到 RREQ 时，建立或更新到源节点的反向路径。这里的更新是指：当所获新路由信息中的相关序列号比原路由表中相应路由的序列号大，或序列号相同而跳数比原来的小时，则改变相应路由的目的序列号或者跳数，并增加路由的生存时间。一旦 RREQ 到达了目的节点或拥有一条通向目的节点足够新的路径的中间节点，这个节点就通过反向路径单播一个路由响应分组（RREP）给源节点。在 RREP 通过反向路径到达源节点的过程中，这条路径上的节点在它们的路由表中建立起通向目的节点的正向路径。如图 2-30 所示，信源 1 与信宿 8 采用 AODV 的路由发现过程。

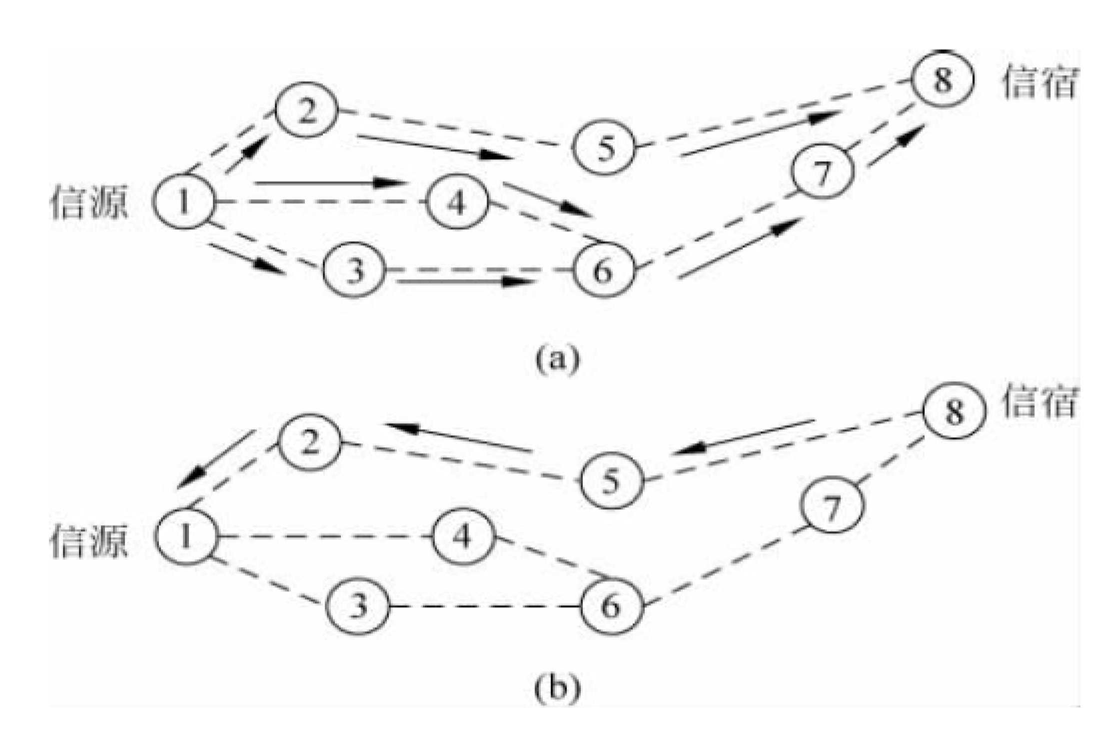


图 2-30　AODV 路由发现

其他路由协议包括 ZRP（zone routing protocol），它是一种利用集群结构、混合使用表驱动和按需路由策略的移动 Ad Hoc 网络路由协议，以及利用地理位置信息的路由协议、QoS 路由协议、功率感知路由协议、多播路由协议等。

2.5.3　协作通信技术

在上一节中讲过利用 MIMO 技术来对抗信道的衰落，提高接收信号的信噪比，虽然 MIMO 技术已成为新一代无线网络的关键技术之一，但它的应用却还有一定的局限性。在某些实际场景中，如在蜂窝通信系统的上行链路中，由于移动终端的体积、成本以及功耗等限制，在终端上安装多天线来实现空间分集增益不易实现，因此促进了协作通信技术的产生与发展。

协作通信技术可以使单一的用户通过分享其他用户的天线，创建一个虚拟的 MIMO

系统，从而获得分集增益，因此协作通信中终端不需要安装多根天线，而是产生一种类似多天线发送的虚拟环境，使之成为 MIMO 技术很好的扩展和补充。

该技术融合了分集技术与中继传输的技术优势，形成了分布式虚拟 MIMO 系统，获得空间分集增益，提高系统传输性能；克服了相干距离等限制，在不增加天线数目的基础上，可在传统通信网络中获得与多天线及多跳传输情况下相近的传输增益。

协作通信技术与多跳中继技术的不同之处在于：在多跳中继技术中，中继节点的唯一目的就是帮源节点发送信息，而协作通信技术中整个系统的资源固定，各用户既可充当中继节点帮助源节点发送信息，又可作为源节点发送自己的信息。

中继信道模型是协作通信技术的基础，根据网络中是否存在空闲节点，存在两种不同的协作通信模型，如图 2-31 所示[17]。

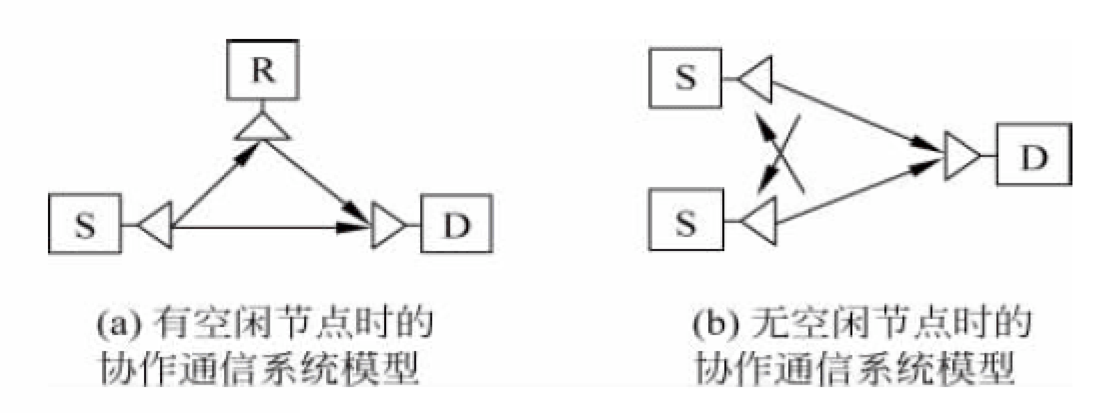


图 2-31　协作通信模型

当系统中不存在空闲节点时，采用图 2-31(b)所示的系统模型，此时用户既要传送自己的信息，又要传送其协作伙伴的信息，会牺牲一部分自己的资源，造成如带宽、发射功率等资源的浪费，降低有效通信数据流量。另一方面，用户也通过协作分集利用了其协作伙伴的空域资源。综合以上两个方面，当协作通信产生的正面效应大于负面效应时，系统会获得相应的性能增益。

当系统中存在空闲节点时，空闲节点可以当作中继节点进行数据转发等功能。

协作通信技术虽说可以有效地增加通信数据流量，提高通信性能，但也存在以下几个方面的问题[18]。

(1) 中继节点带来的干扰问题。协作通信系统中的多数研究停留在已定节点，如何进行协作以实现空间分集的问题。中继节点的选择也是一个非常重要的问题，即资源和合理分配问题。如果中继节点选择不当，不仅会增加信息丢失的概率，还会对其他信源、目的节点间的通信产生干扰，从而降低整个网络的性能。

(2) 通信的安全保密性问题。因为协作通信中的传输模式是从源节点到中继节点、中继节点再到目的节点的传输方式，因此信息的传输、交付机制变得更加复杂，通信的保密性问题也变得更为重要。

(3) 信道问题。在现有的协作通信研究中，对信道的研究主要集中在平坦衰落信道，而现实条件下的信道条件异常复杂，这方面还没有足够的研究。

2.6　移动终端的融合

移动终端又称为移动通信终端，指在移动通信中使用的计算机设备，包括手机、笔记本、平板电脑、车载电脑等。

移动终端具有以下几方面的特点：在硬件上，具有中央处理器、存储器、输入/输出设备等，即为具有通信功能的微型计算机；在软件上，移动终端必须具备操作系统，如Android、IOS、Windows Mobile等；在通信能力上，移动终端必须有灵活的接入方式及高速网络切换的能力，可以支持GSM、WCDMA、CDMA2000、TDSCDMA、LTE、WiMAX等多种制式的网络；在功能上更加强大，更加注重人性化、个性化、多功能化软件的开发。

异构无线网络的融合与互通需要多模终端的支持，多模终端拥有多个无线接口，能够接入不同的无线网络。因此，设计异构多模终端接入选择的功能架构和接入选择策略，保持用户始终接入最合适的网络，有效利用全网的无线资源，在研究异构无线网络融合方面，是重要且颇具挑战性的研究课题。

软件无线电作为一种具有开放式结构的新技术，是将模块化、标准化的功能单元经过一个通用的硬件平台，利用软件方式实现多种无线通信系统的接入，是解决移动终端在不同系统中接入的关键技术。

下面分别对多模移动终端和支持软件无线电的终端进行介绍。

2.6.1 多模移动终端

多模终端就是指能够支持多于一种的无线接入技术，并能够通过这些无线接入技术获得服务的终端。根据国际标准3GPP TR21.910的多模终端分类定义，多模终端分为以下4类[19]：

（1）手动模式变换多模单待终端。此种模式中终端的变化只能通过手动选择来进行，终端可以支持多种接入技术，但只有一种接入技术处于使用状态。

（2）自动模式变换多模单待终端。此模式中终端自动完成不同接入网络之间的切换，只有一种接入技术处于使用状态，与此同时终端还可以对其他网络进行测量和监听，并上报其目前工作的模式。

（3）双收单发多模双待终端。此模式中终端可以同时接收两种接入网络的信号，但只能在一种模式下进行信号发送。

（4）双收双发多模双待终端。与上一种模式的终端不同的是，此种终端可以在多种网络模式下同时接收和发送信号，但需要考虑不同网络间的干扰性。

现阶段大多数多模终端采用自动模式变换多模单待模式或双收双发多模双待模式。

例如，LTE多模双待手机是指LTE模式与手机支持的另一种模式（2G或3G模式）同时待机，LTE模式提供数据业务，而2G/3G的电路域提供话音业务，当手机不在LTE覆盖区域时，数据业务也可以由2G/3G的分组域提供。

多模终端的架构由射频前端、射频芯片、模拟基带芯片、数字基带芯片以及电源管理模块几部分组成。LTE/3G/2G三模双待终端架构参考示意图如图2-32所示。

在此多模双待终端架构中，LTE与其他两种模式之间是相互独立的，支持不同模式的同时收发，射频前端、射频芯片以及数字基带芯片都独立成套，每一套芯片组都能独立进行工作，可以理解为两部能够同时工作的不同模式的终端，唯一不同的是两部终端的射频部分离的较近，不同模式发射和接收会造成相互之间的干扰，需要天线具有很好的

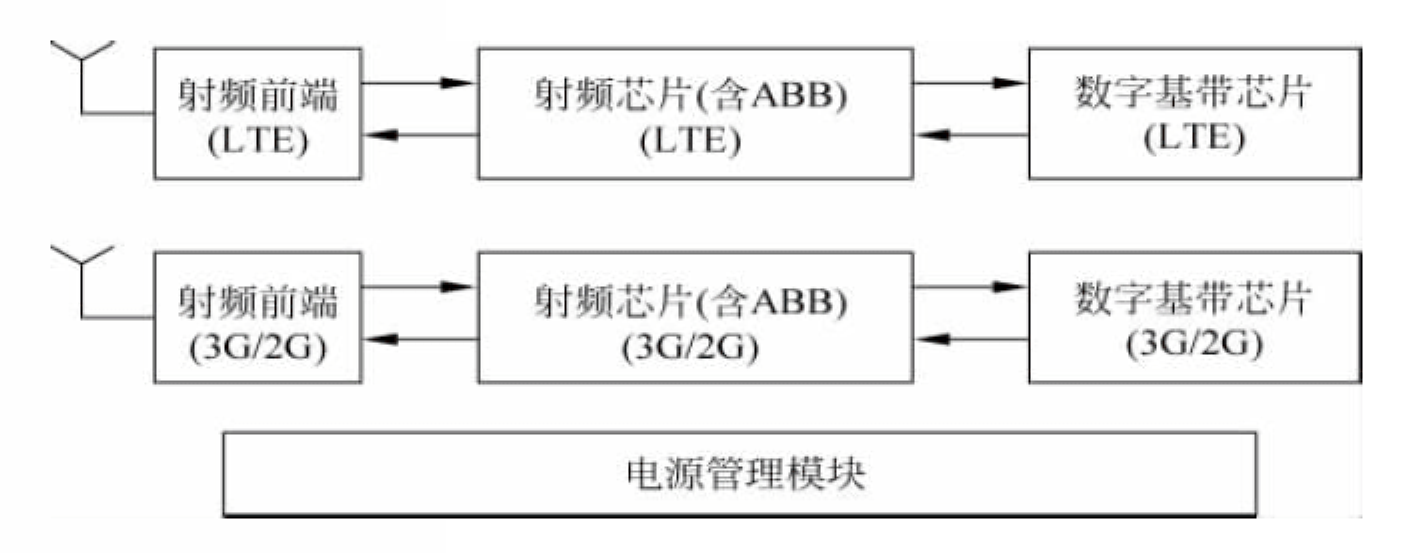


图 2-32　LTE/3G/2G 三模双待终端架构参考示意图

隔离度才能满足多模收发的射频性能要求。

2.6.2　支持软件无线电的移动终端

支持软件无线电技术的终端是通过硬件和软件的结合使终端具有可重配置能力，并实现多模式、多频段、多功能的无线通信能力。软件无线电（SDR）的关键在于通过使 A/D、D/A 尽可能靠近射频端，从而使信号尽早地数字化，将过去使用硬件才能完成的工作用软件来代替。然后借助可编程器件（DSP/FPGA/CPU 等）强大的信号处理功能和灵活的可重构特性实现多种通信标准的支持。

根据软件在软件无线电中的应用程度，可将终端应用软件无线电分为 5 个等级[20][21]：

第一级为硬件无线电（digital hardware radio，DHR）。通过开关、拨号盘及其他按钮来完成操作，仅支持单一或非常局限的功能，不存在软件控制功能的改变，如第一代模拟制式手机 MOTO-3200。

第二级为软件控制无线电（software controlled radio，SCR）。软件实现有限的控制功能，其射频前端电路、中频和基带部分由不同的芯片完成，但可以通过软件控制通信路径。如最早的同时支持 PHS 与 GSM 的双模手机。

第三级为软件定义无线电（software defined radio，SDR）。软件定义无线通信协议，频带、空中接口协议和功能都可通过软件下载和更新来进行变换，因此不用对硬件进行更新。该平台一般基于模块化、开放性的架构，射频、中频、基带部分由不同的模块构成，基带部分一般采用 FPGA/IX、P/CPU 等，通过配置不同的软件来支持 2G/3G/LTE/WiMAX 等多种制式。现在基于软件无线电的终端设备便处于第三级的水平，即软件定义无线电。

第四级为理想的软件无线电（ideal software radio，ISR）。在接收端或发送端无须进行任何下变频或上变频转换，将 RF 前端的输出直接接入 ADC，然后进入到 DSP，消除了大部分的模拟部件，从而降低了失真和噪声。

第五级为终极软件无线电（ultimate software radio，USR）。这种无线电没有天线、频率及带宽的限制，通过可编程能支持各种频率和功能，并快速地完成空中接口间的检测和转换。

理想的终端软件无线电结构如图 2-33 所示。它由射频天线、噪声放大器、A/D 与

D/A 转换、通用处理器等功能单元组成，实现一个模块化、开放性、便于灵活扩展和重复利用的可编程硬件平台。天线接收到的模拟信号经过模拟部分处理之后，通过 A/D 转换器进行数字化，随后可编程基带根据不同的无线环境来处理数字化信号。其各个单元功能如下所示。

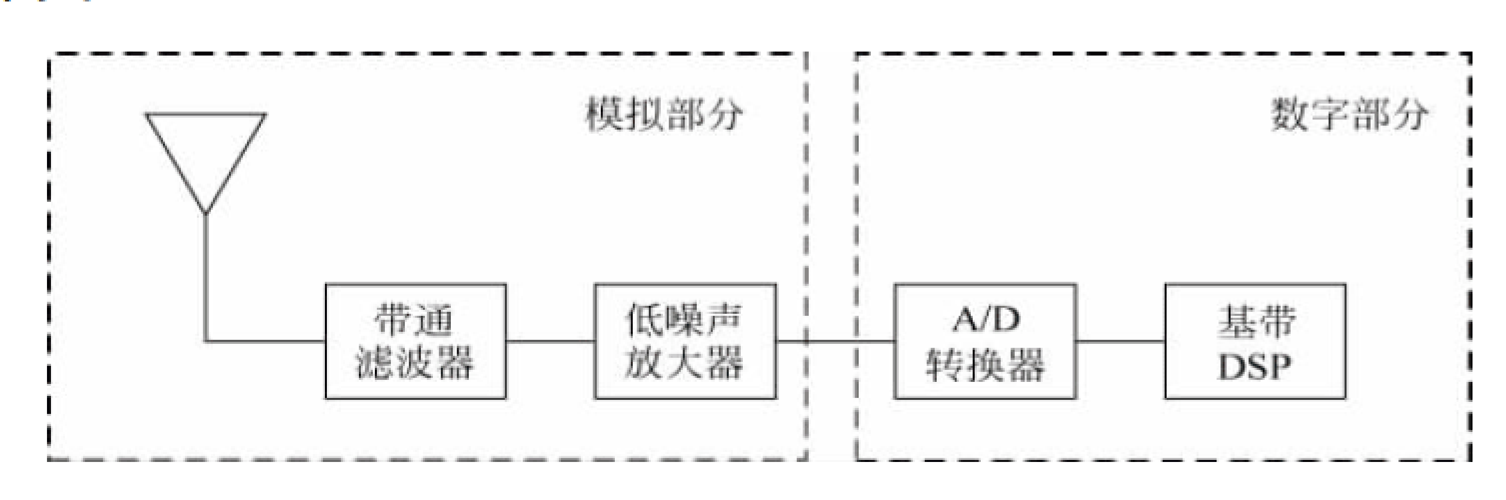


图 2-33 理想终端软件无线电的设计

(1) 多频段天线：软件无线电要支持多种通信标准/制式，因而要在很宽的频宽内工作，目前大多采用组合式的多频段天线。

(2) 射频前端：射频前端包括低噪声放大器、功率放大器及滤波器等，其工作频率范围应当足够宽。一些网络如 LTE 等对灵敏度和 BER 要求更严格，这也对 LNA/PA 的线性度、功耗、噪声提出更高要求。

(3) A/D 与 D/A 转换：目前大多数无线通信标准都工作在 VHF、UHF、SHF 频段，直接在射频上进行 A/D 和 D/A 变化还难以实现，一般是通过混频器完成模拟信号的上下变频。ADC 和 DAC 的采样率/更新率、分辨率、动态范围(SFDR)、抖动等都是影响 SDR 性能的关键因素。

(4) DSP/FPGA/GPP：DSP/FPGA/GPP 都是实现 SDR 的主流处理硬件，DSP 的处理速度低但适于复杂算法的实现，对于数字信号处理非常擅长；FPGA 的工作速率高但复杂计算能力相对较弱，适用于复杂性不高的计算密集型任务；GPP 难以保证算法实时性要求但通用性和灵活性高，因此在实际的应用中会按它们各自的特点组合使用。

参考文献

[1] 蒋群艳. 全 IP 无线异构网络融合及其切换研究[D]. 上海：上海交通大学电子工程系，2007.

[2] Ahmavaara K，Haverinen H，Pichna R. Interworking architecture between 3GPP and WLAN systems[J]. IEEE Communications Magazine，2003，41(11)：74-81.

[3] 纪阳，阮征，张平. 基于全 IP 的 3GPP-WLAN 网络互通体系结构[J]. 电信工程技术与标准化，2004(3)：51-54.

[4] C-RAN 无线接入网绿色演进白皮书[R]. 中国移动研究院，2010(8).

[5] Hadzialic M，Dosenovic B，Dzaferagic M，et al. Cloud-RAN：innovative radio access network architecture[C]//ELMAR，2013 55th International Symposium. IEEE，2013：115-120.

[6] Zaki Y，Zhao L，Goerg C，et al. LTE mobile network virtualization[J]. Mobile Networks and Applications，2011，16(4)：424-432.

[7] Xiang-Nan D U，Hua-Ming F U. Research on SIP architecture[J]. Information Technology，2008. 03. 033：75-78.

[8] Mahonen P, Riihijarvi J, Petrova M, et al. Hop-by-hop toward future mobile broadband IP[J]. IEEE Communications Magazine, 2004, 42(3): 138-146.
[9] 孙海波. 移动 MPLS 技术的研究与性能分析[D]. 南京: 南京邮电大学, 南京邮电学院, 2004.
[10] 林笛. 基于 MPLS 技术的移动 IPv6 服务质量研究[D]. 天津: 天津大学电信学院, 2005.
[11] 肖长水, 姒茂新, 沈萍萍. 移动 IPv6 切换技术综述[J]. 计算机应用与软件, 2010, 27(4): 158-161.
[12] Bianchi P, Loubaton P. On the Blind Equalization of Continuous Phase Modulated Signals Using the Constant Modulus Criterion[J]. IEEE Transactions on Signal Processing, 2013, 55(3): 1047-1061.
[13] 程卫军. 无线通信网络中多跳中继技术及其应用的研究[D]. 北京: 北京邮电大学电子工程学院, 2004.
[14] 詹瑞, 刘俊. 协作通信技术的研究进展[J]. 数字通信, 2013, 40(1): 43-47.
[15] 由磊, 宋俊德, 彭海兰. 多跳中继增强的 WiMAX 网络架构与组网成本分析[J]. 移动通信, 2008, 32(8): 28-32.
[16] 张波. 移动 WiMAX 系统的中继选择和切换机制研究[D]. 济南: 山东大学通信与信息系统, 2010.
[17] 梅中辉, 李晓飞. 协作通信技术[J]. 数据通信, 2009(5).
[18] 詹瑞, 刘俊. 协作通信技术的研究进展[J]. 数字通信, 2013, 40(1): 43-47.
[19] 朵灏, 刘臻, 田云飞. LTE 多模终端技术研究[J]. 电信网技术, 2012(4): 38-42.
[20] Wireless innovation forum[EB/OL]. http://www. wirelessinnovation. Org/.
[21] 张鹏. 软件无线电技术在移动通信测试领域的应用[J]. 电子测量技术, 2013, 36(3): 110-117.

第3章 异构无线网络融合关键技术

本章首先阐述无线网络的移动性管理技术，对电路域和分组域的移动性管理技术分别进行分析，并提出自适应位置管理方案和鲁棒移动性管理方案；接下来对异构无线网络资源管理的三种模型以及三种联合接纳机制进行详细分析，并针对多接入网络选择过程中的网络构架和接入选择算法进行简要说明；最后讨论了提高异构网络 QoS 的三种服务模型，并对 QoS 映射分类进行探讨。

3.1 异构无线网络移动性管理

移动性管理是随着移动通信系统的产生而出现的，它是移动通信系统正常运转的关键。所谓移动性管理，就是指通信网络能够定位漫游终端，并且传递呼叫的能力，以及用户在网络内部的不同区域（位置区）之间进行移动的过程中能够保证正在进行的通信的连续性。其中包括越区切换、寻呼、位置登记和更新等操作。位置管理和切换管理是移动性管理的两个基本问题，位置管理是指移动终端周期性或非周期性地向网络更新它们的位置信息；切换管理是指正在进行的呼叫或会话从一个物理信道转换到另一个物理信道的过程。

在异构无线网络环境下，移动性管理的目标更加多样化，根据移动性管理所支持目标的不同，可以分为终端移动性、个人移动性、会话移动性、子网移动性和业务移动性几类[1][2]。

终端移动性指用户或终端在移动的过程中，始终具备通信能力及使用业务的能力，不受接入技术变化的影响，并且网络始终可以识别并定位到该终端的具体位置。

个人移动性指同一用户基于个人标志不论使用何种终端都具备通信及访问业务的能力，即在不同位置、通过不同的接入网络依然能实现通信的能力。

会话移动性指当终端移动并改变接入网络时，通信仍能保持不中断的能力，强调通信的连续性。

业务移动性强调业务的支持能力，具有跨地区、跨运营商、跨终端业务可携带性的能力，即用户不论在何种位置，接入何种网络，使用何

种终端，均可以无差别地体验相同业务的能力。

子网移动性指当一组用户作为一个整体时，若改变其网络的附着点仍具有通信的能力，如航海中的游轮、火车及汽车上的网络等。

3.1.1 移动性管理简介

1. 位置管理

位置管理实质上包括两方面的过程，如图 3-1 所示。第一是位置注册（又称“位置更新”），在这个阶段，一方面移动终端将自己的接入点位置信息周期性或非周期性地通知网络，另一方面，网络可以对用户进行鉴权，并修改用户的位置文件。第二是呼叫传递，即网络通过查询，得到用户的位置文件和移动终端当前的位置。

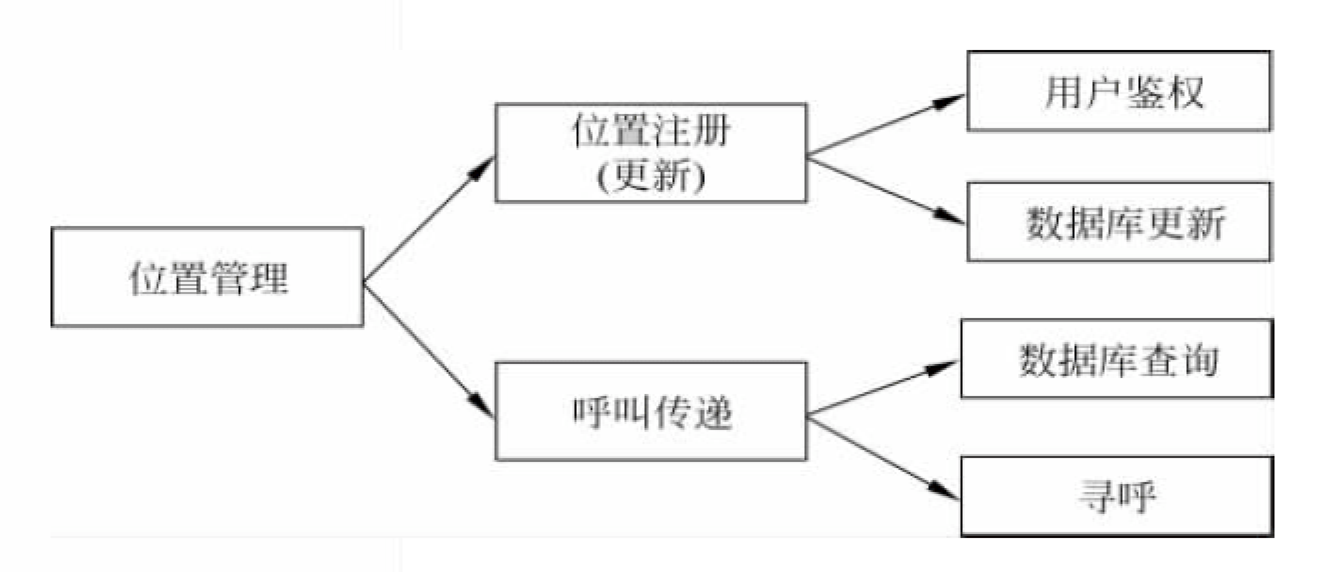


图 3-1 位置管理操作

位置管理研究包括几方面的内容：数据库结构的设计，动态数据库更新，减少查询时延，实体间的信令传输，移动终端的寻呼方法，控制寻呼时延和安全性等。当移动终端的数量大大增加时，需要优化位置管理方案来提供更有效的位置管理[3]。

2. 切换管理

切换管理分三步，如图 3-2 所示。第一步是切换发起。当用户发生了移动或网络条

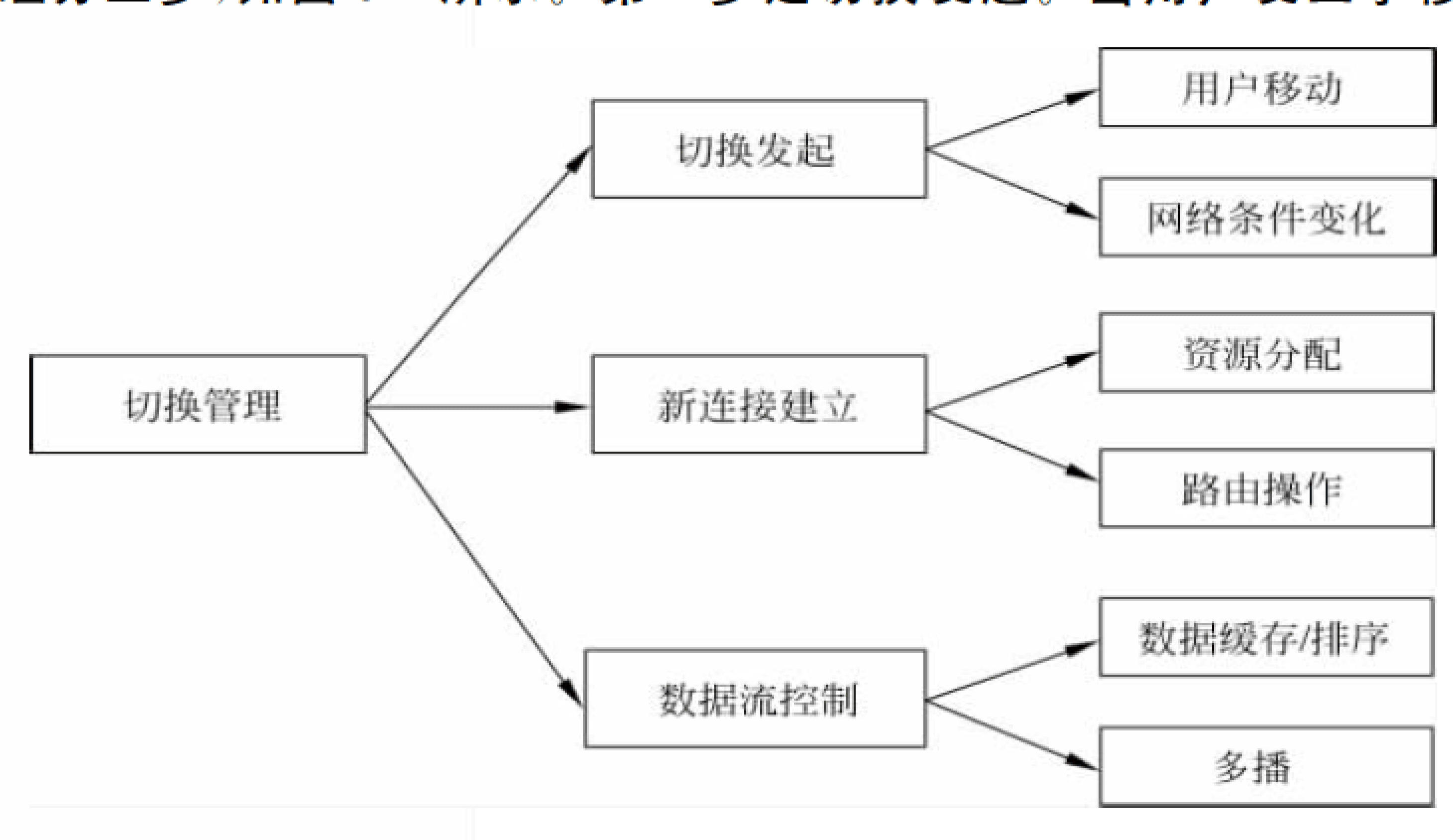


图 3-2 切换管理操作

件发生了变化，切换可以由用户或者网络发起。第二步是建立新的连接。在网络控制的切换（network controlled hand over，NCHO）和终端辅助的切换（mobile assisted hand over，MAHO）中，由网络来发现新的资源并执行路由操作，最终建立一个新的连接。而在终端控制的切换（mobile controlled hand over，MCHO）中，是由移动终端来发现新的资源，在网络允许的条件下建立新的连接。第三步是数据流控制。在这个阶段，根据协定的业务质量保证，数据从旧的连接通道转移到新的连接通道。

根据切换期同时涉及的基站数量分类，切换可以分为软切换和硬切换。在切换过程中，如果移动终端可以同时与多个基站保持连接，并使用一定的分集技术对多个信号进行合并，这种切换称为软切换。反之，在切换过程中，如果移动终端在一个时刻只与一个基站保持连接，这种切换称为硬切换。

按切换前后的接入类型分类，移动终端的切换又可以分为两类：水平切换（horizontal handover）和垂直切换（vertical handover），如图 3-3 所示。水平切换和垂直切换的区别如表 3-1 所示。

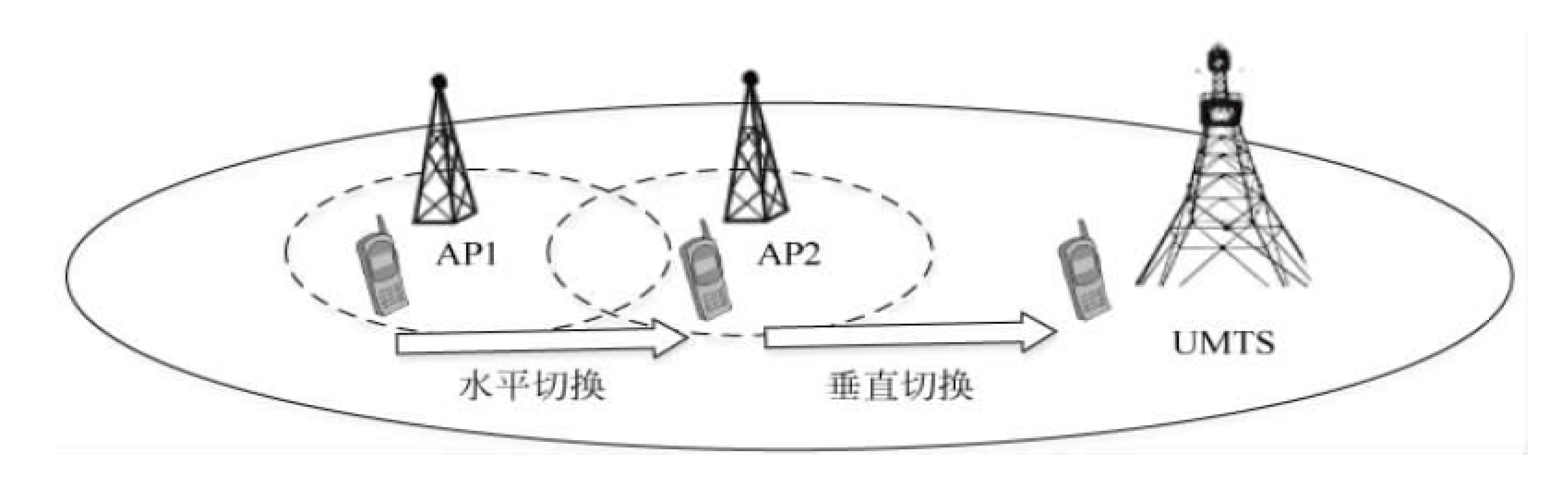


图 3-3　垂直切换与水平切换

表 3-1　水平切换和垂直切换比较

切换类型	切换原因	切换判决因素	切换发起方
水平切换	物理位置变化	信号强度、信道可用性等	网络发起
垂直切换	物理位置变化/接入技术变化	只基于水平切换的因素决策是不够的，还需考虑 QoS、用户偏好服务资费等多方面因素	用户可以根据偏好设置或者 QoS 的考虑，主动发起切换

传统的切换主要是水平切换，移动终端通常在属于同一种无线接入技术的基站覆盖范围内，移动终端的通信只需要在同种无线接入技术的多台基站间转接，切换前后的无线接入点类型相同。一般来说，水平切换中通常只有一个无线网卡参与了切换过程，也称为单模切换。

“垂直切换”与水平切换相对应，是指终端在不同的无线系统之间移动中完成的切换，是相对于水平切换而言。切换前后的无线接入点类型不同，在这种切换中涉及不同的无线网卡，但是某一时刻仅有一个接口进行数据传输，也称为多模切换。垂直切换又分为“上行”切换和“下行”切换。“上行”切换是指切换到一个更大覆盖范围以及较低的接入速率的网络，而“下行”切换是指切换到覆盖面积区域较小但接入速率较高的网络。以图 3-3 为例，MN 在 AP1 和 AP2 之间的切换称为水平切换，MN 从 AP1 或 AP2 切换到 UMTS 网络就为垂直切换。因为 WLAN 覆盖范围小于 UMTS，接入速率大于

UMTS，因此，从 AP1 或 AP2 到 UMTS 的切换就称为上行切换，反之就为下行切换。

3．水平切换判决算法

水平切换性能的提高可以借鉴传统的蜂窝移动通信系统的切换算法。在传统的蜂窝移动通信系统中，切换动作在移动节点跨越不同基站的小区时发生，所使用的参数通常是移动终端上测量的接收信号强度（RSS）、信噪比（SIR）和信道质量（误码率、丢包率等）。通过比较不同基站的这些参数值，切换判决机制[4]，选择一个合适的基站作为切换目标。网络层水平切换，需要添加能够及时探测底层链路状态的机制，通过该机制收集到的不同无线接入点无线链路状态的差异，进行切换判决操作。

以蜂窝移动通信系统为例，切换主要发生在移动台跨越不同蜂窝小区之间，作为公共的信令协议得到广泛的研究。传统的水平切换控制机制主要采用接收信号强度（RSS）作为接入网络可用性的指示，并由此判断切换是否发生。典型的切换判决算法有以下几种：

（1）基于接收信号强度法。如果接入网络附着点的 RSS 高于当前的附着点的接收信号强度 $\left(RSS_{\text{New}}>RSS_{\text{Current}}\dfrac{n!}{r!(n-r)!}\right)$，将发生切换。

（2）基于接收信号强度加门限法。如果接入网络的接收信号强度 RSS 高于当前接入网络的接收信号强度，且当前接入网络的接收信号强度 RSS 比预定义的门限值 T 还要低时（$RSS_{\text{New}}>RSS_{\text{Current}}$ 且 $RSS_{\text{Current}}<T$），将发生切换。

（3）基于接收信号强度加滞后余量法。如果接入网络的接收信号强度高于当前接入网络的接收信号强度，且高出量大于预定义的门限 H（$RSS_{\text{New}}>RSS_{\text{Current}}+H$），切换将发生。

另外，为了减少乒乓切换，上述算法中有时设定一个切换定时器（timer）。当切换判决开始时，切换定时器开始计时，如果在定时周期到期，条件仍然得到满足，切换将发生。

4．垂直切换判决算法

在异构网络环境中，由于不同网络间网络特征差异较大，传统意义上的水平切换控制机制基于接收信号强度的比较，并不适合垂直切换的要求。因为垂直切换发生在不同无线接入链路之间，由于无线接入技术的差异，底层链路的参数在取值范围、意义等方面完全不同，因而不同链路的参数之间不具备可比性。垂直切换过程主要包含以下三个步骤[5]：

（1）切换发起。无线信号强度的下降是切换发起的主要因素，除此之外，还需要考虑网络的负载均衡、QoS、用户偏好、服务费用等因素。切换发起阶段的主要任务是移动终端搜索和发现当前所处环境下的可接入网络。

（2）切换判决。切换判决要完成的主要任务是移动节点对可接入网络进行估计，然后做出切换决策，最后选择最优的切换目标网络。

（3）切换执行。若移动节点决定执行垂直切换后，通信将被转移到新的链路，与此同时原来的信道将被释放。

在系统搜索阶段，具有多无线技术接口的MN必须接收不同无线技术网络发送的服务公告，搜索发现可接入网络的最简单办法就是保持移动节点的多接口都处于激活状态，但是这样会消耗大量功率。为此，可以改进使MN周期性检查当前位置以及可接入的无线网络列表，只有当MN发现已经移动到原服务网络之外的某一网络覆盖范围之内时，才激活相应接口。

在研究异构网络融合中，面临的主要挑战之一就是设计高效、合理的垂直切换判决算法，以满足不同网络间无缝移动性管理的要求。目前，业内针对垂直切换判决的研究，可以从以下几个方面展开：

(1) 基于信号强度比较的判决策略。

(2) 基于简单加权SAW的方法。

(3) 基于人工智能和模糊逻辑的判决策略。

(4) 基于代价函数的判决策略。

3.1.2 电路域的移动性管理

1. 现有的电路域移动性管理技术

现代移动通信系统始于20世纪20年代，从早期的专用移动通信系统到80年代趋于成熟的数字蜂窝移动通信系统，目前已经进入了第四代数字移动通信系统的商用阶段。移动通信的主要目的是为用户提供个人通信业务(PCS)，即指各种各样的无线接入及通过一个小型终端提供的个人移动业务，从而达到在任何时间、任何地点和以任何形式通信的目标。常见的PCS技术包括蜂窝移动通信技术和无线电话技术。

针对蜂窝移动通信中的电路域，目前存在两个通用的移动性管理标准，即IS-41和GSM MAP(移动应用部分)。IS-41主要用于北美的AMPS(advanced mobile phone system)、IS-54、IS-136和PACS(personal access communication system)，而GSM MAP主要用于欧洲的GSM、DCS-1800和DCS-1900。

1) IS-41标准

目前通用的网络接口标准主要为IS-41标准，又称为ANSI41标准，这套标准是由ANSITIA授权LUCENT公司贝尔实验室制定的，最先发布于1988年[6]。由于北美通信系统的一个最主要的特点就是平滑过渡，于是产生了一系列不断发展的标准体系。IS-41系列标准在其发展过程中主要产生了IS-41A、IS-41B、IS-41C、ANSI-41、IS-41D等几个版本，与IS-41B有关的TSB文件有TSB41、TSB51、TSB55、TSB56、TSB64、TSB65等，其中除TSB56以外其余均已在1996年修订并加入到IS-41C版本，使之成为一个相对完善的版本，不久之后，美国国家标准委员会接收其为正式的网络接口标准，称为ANSI-41标准。

与所熟悉的GSM系统一样，CDMA系统的网络部分由接入网和核心网两部分组成，而核心网部分主要由数据库系统(HLR)和交换机系统(MSC)等网元构成。

IS-41标准规定了这些网元之间(主要是HLR与MSC之间)的操作，主要用于核心网部分提供服务，如用户识别与鉴定、呼叫路由等。因此CDMA核心网也通常被称作

IS-41 网络。IS-41 网络主要由以下网络实体及其相关接口构成，如图 3-4 所示。

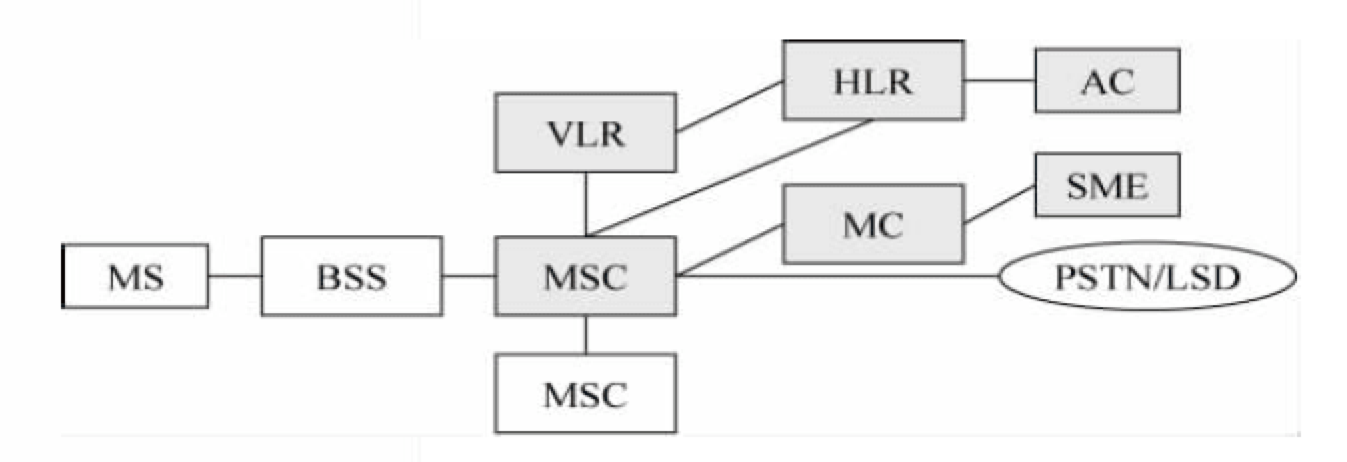


图 3-4　IS-41 网络实体及其相关接口

(1) MSC，移动交换中心，无线网络与其他公共交换网的用户数据接口或其他无线网络之间的业务数据(语音/短信)交换点。

(2) HLR，归属位置注册服务器，记录每个用户的相关信息，主要包括当前用户位置信息和用户相关信息与参数。HLR 能够提供用户移动性管理及业务控制服务；HLR 一般同时管理多个 MSC，是移动用户管理的静态数据库。

(3) VLR，拜访者位置寄存器，为 MSC 提供用户号码、本地用户服务等参数。VLR 为漫游到本地的拜访者提供移动性管理服务，具有写入和删除的功能，是一个动态用户数据库。VLR 一般与 MSC 放在一起，并可能为多个 MSC 进行服务。

(4) AC，鉴权中心，存储保证移动用户通信不受侵犯的鉴权参数等信息，如随机数、密钥等功能实体，AC 可能会内置在 HLR 中。

(5) MC，短消息中心，提供短消息的存储转发功能，也可以提供短消息补充业务。

(6) SME，短消息实体，可以发送和接收短消息的实体。SME 可能位于一个固定的网络，如 HLR、MC、VLR、MS 或 MSC 之中。

而 IS-41 网络主要提供的业务有自动漫游业务、语音呼叫业务、位置寻址与翻译和短消息业务等。

2) GSM MAP 标准

在 GSM 数字蜂窝移动通信网络结构中，移动业务交换中心 MSC 和访问位置寄存器 VLR、归属位置寄存器 HLR 之间采用的是 NO. 7 信令系统。GSM 标准在使用 NO. 7 信令低层协议的基础上，同时制定了高层应用协议。具体来说，如图 3-5 所示，它在 SCCP (信令连接控制部分)之上增加了两层实体：事务处理应用部分 TCAP 和移动应用部分 MAP[7]。

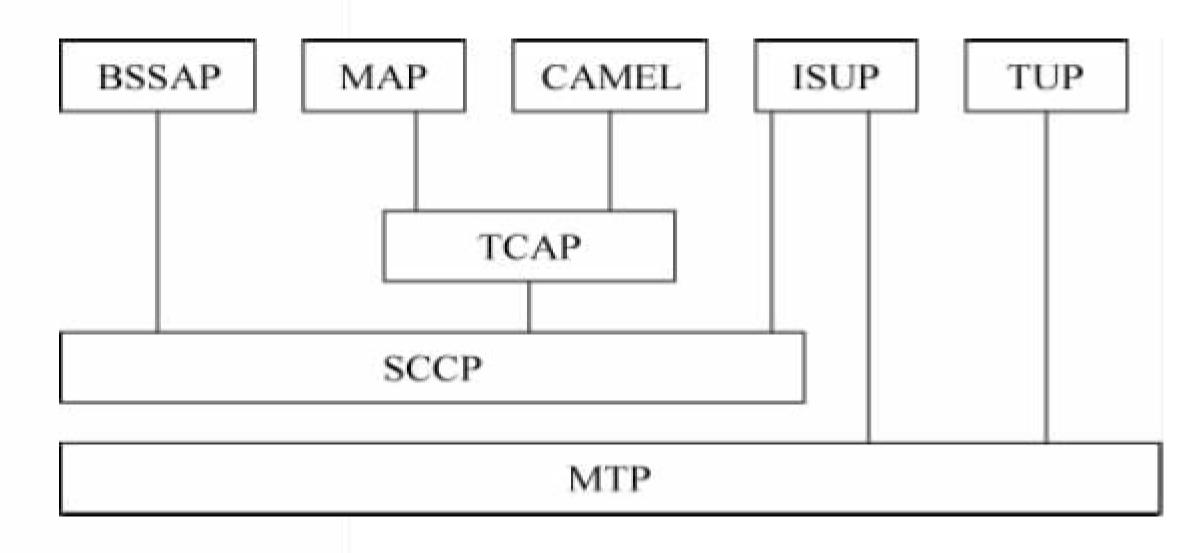


图 3-5　7 号信令功能模块

TCAP为网络的广泛应用提供了支持，并为这些应用消息在网络上的传输提供了保证，MAP则是专门针对GSM的要求而设计的，它和TCAP同属于OSI模型的第7层中。MAP使用TCAP提供的服务，为不同功能实体之间提供端到端的通信连接，定义了移动通信核心网的各网元间为了实现移动台的自动漫游功能而进行的信息交换方式。

2. 电路域的位置管理

不管IS-41还是GSM MAP，它们的位置管理策略非常类似，都采用了两层数据库的结构。其中，涉及两种类型的位置信息数据库，即家乡位置寄存器HLR和访问位置寄存器VLR。通常情况下，每个网络有一个HLR，注册业务类型和位置信息等都永久地保存在注册网络的HLR中。而VLR的个数和部署，在不同的网络中有比较大的差别。VLR中保存了从HLR下载的访问移动终端的相关信息。位置管理中的位置注册和呼叫处理功能都需要经过一定的信令交换才能完成。在蜂窝移动通信中，负责信令交换的是7号信令系统(SS7)，图3-6中显示了连接HLR、VLR和MSC的7号信令网[8,9]。其中，STP是信令转换点，主要负责信令消息的路由，一般采用双机备份，MSC为移动交换中心。

位置管理包括位置注册过程和呼叫传递过程。下面分别介绍这两个方面。

1) 位置注册过程

当移动终端进入一个新的LA(位置区域)，就执行位置更新操作向网络报告最新的LA位置信息。通常情况下，一个LA由多个小区构成，属于同一个LA的BTS(基站收发台)都连接到相同的MSC。

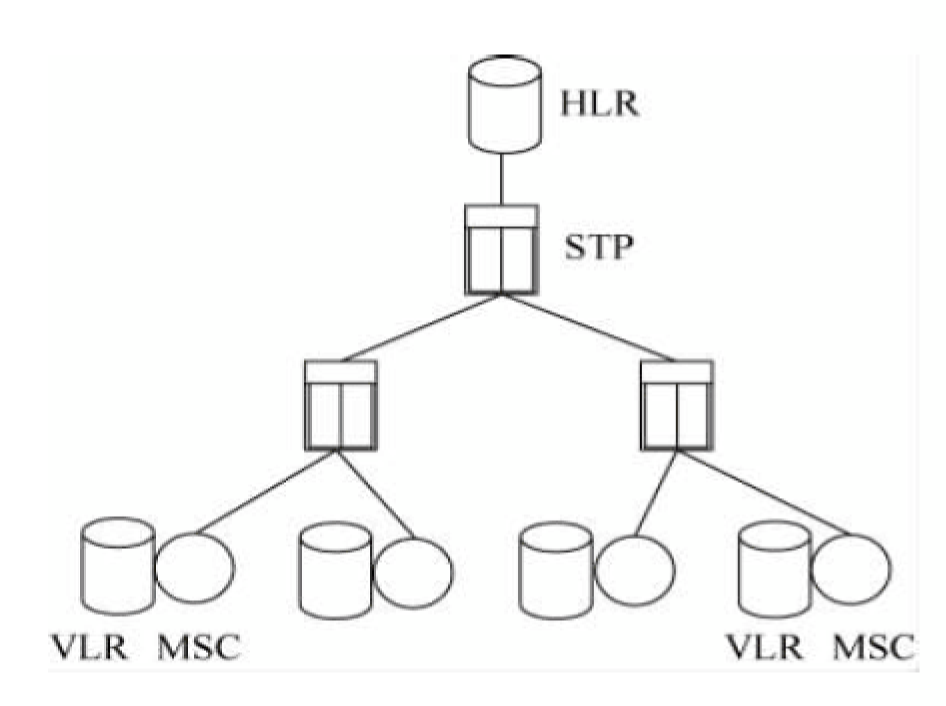


图3-6 7号信令网

当移动终端(MT)进入一个新的LA时，存在两种情况。第一种情况是，新LA与原LA同属于一个VLR。那么，只需要更新VLR中的LA记录即可。第二种情况是，新LA与原LA属于不同的VLR。那么，需要进行下列操作：

(1) 向新VLR注册MT。

(2) 新VLR发送注册消息给MT的HLR，HLR鉴权MT并完成VLR记录的更新。

(3) HLR向旧VLR发送注册取消消息。

(4) 旧VLR删除MT的相关记录，并发送取消确认消息给HLR。

2) 呼叫传递过程

呼叫传递主要包括两个步骤，首先要确定被叫MT的服务VLR和MSC，然后确定被叫MT的当前小区位置。在确定了MT的具体位置后，主叫MSC就可以通过7号信令网与被叫MSC建立呼叫连接。

当主叫MT发起一个呼叫到被叫MT时，按照如下步骤实施：

(1) 由主叫MSC确定被叫MT的HLR，并发送相应的位置请求消息给HLR。

（2）HLR 通过查询，可以得到被叫 MT 的服务 VLR，并向该 VLR 发送路由请求消息。

（3）VLR 收到路由请求消息，则转发给被叫 MT 的服务 MSC。

（4）服务 MSC 给被叫 MT 分配一个 TLDN（临时本地电话号码），并向 HLR 返回带有 TLDN 的回复消息。

（5）HLR 将相关信息转发给主叫 MSC。这样，被叫 MT 的服务 VLR 和 MSC 就确定了。

通常情况下，一个 MSC 和一个 LA 相对应，由于一个 LA 中包含若干个小区，所以，仅仅确定被叫 MT 的 MSC 还不能完全定位 MT。在当前的蜂窝移动通信系统中，确定被叫 MT 的当前小区位置是通过寻呼实现的。LA 中的所有小区通过广播寻呼消息来查找被叫 MT，MT 收到寻呼消息后，向服务 MSC 发送一个应答消息。这样，MSC 就确定了被叫 MT 的当前小区位置。可以看出，如果 MT 的数量增加，那么，寻呼消息的大量增加会消耗很多无线带宽资源。

第三代无线网络由于分成了电路域和分组域，因此，针对这两个区域引入了不同类型的管理区域的概念，电路域中仍沿用 LA，对于分组域则引入了 RA（路由区域），3G-MSC/VLR 使用 LA 来寻呼终端，3G-SGSN 使用寻找 RA 来寻呼终端。关于分组域的移动性管理将在下一小节详细介绍。

3. 电路域的切换管理

从越区切换的控制策略上讨论，在高级移动电话系统 AMPS 中，切换的发起是由网络决定的（即 NCHO）。信号能量的检测是由基站来完成、由 MSC 来管理。每个基站连续地监视它的所有反向语音信道的信号能量，以决定每一个移动台对于基站发射台的相对位置。为了检测小区中正在进行的呼叫的 RSSI，要用基站中备用的接收机，即定位接收机，来决定相邻基站中的移动用户的信号能量，并且将所有的 RSSI 值传给 MSC。MSC 根据每个基站定位接收机接收到的信号能量数据，来决定是否进行切换。

DAMPS 在业务信道上引入了 TDMA 技术，是一种混合类型空中接口的第二代无线通信技术。它对于切换过程所做的一项改进是使移动台也参与切换过程。引入了移动台辅助的切换（MAHO）。每个移动台检测从周围基站中接收到的信号能量，并且将这些检测数据连续地回送给当前为它服务的基站。当从一个相邻小区的基站中接收到的信号能量比当前基站的高出一定电平或是维持了一定的时间时，就准备进行切换。MAHO 方法使得基站间的呼叫切换比在第一代模拟系统中的快得多，因为切换的检测是由每个移动台来完成的，这样 MSC 就不再需要连续不断地监视信号能量。

使用 TDMA 技术的第二代无线网络系统采用了 MAHO 的机制。从移动终端与新基站的连接和与旧基站链接的释放的先后顺序上看，基于 TDMA 的第二代蜂窝通信系统和第一代蜂窝通信系统的切换操作都是在切换时给用户分配一个不同的无线信道，并且是“先断开后连接”的完成无线链路的转换，即“硬切换”。但是采用 CDMA 技术的第二代系统，可以采用“先连接后断开”的“软切换”机制，因为扩频通信用户在每个小区里都共享相同的信道。因此，切换就不意味着所分配的信道在物理上的改变，而是由一个不同的基站来处理无线通信任务。通过同时计算多个基站接收到的一个用户的信号，MSC 就可能及时地判断出任意时刻用户信号的哪种“版本”是最好的。CDMA 中将切换

分成软切换、更软切换和硬切换。

为了覆盖和负载平衡的原因，3G 需要支持 WCDMA 与 GSM 系统间的切换。在 WCDMA 配置初期，可能采用"热点"覆盖，所以有必要能够切换到 GSM 系统以提供连续的覆盖，从 GSM 切换到 WCDMA 可用来减少 GSM 小区的负载。当 WCDMA 网络的业务量提高时，由于负载的原因而进行的双向切换是很重要的。

3.1.3 分组域的移动性管理

1. UMTS 网络架构

通用移动通信系统 UMTS 网络结构如图 3-7 所示。它可以分成两个基本部分：无线接入网络(radio access network，RAN)和核心网(core network，CN)，根据其业务功能的不同，核心网又可以分为电路域和分组域。分组域实体功能如下：

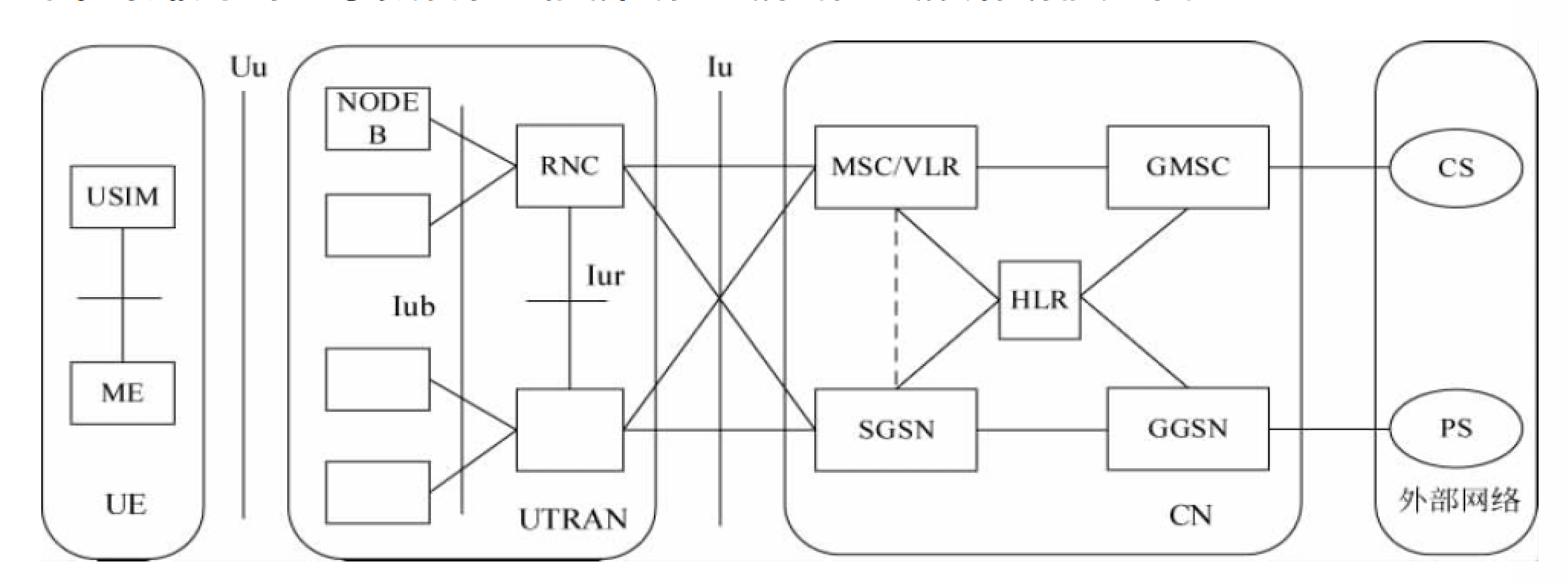


图 3-7 UMTS 网络结构图

(1) SGSN 主要负责为 MS 提供分组数据的传送和接收，进行路由选择，建立移动用户到 GGSN 之间的传输通道。此外，SGSN 还负责 MS 的移动性管理、鉴权、定位和识别移动台的状态并收集与移动终端有关的关键呼叫信息。

(2) GGSN 充当 UMTS 核心网与公共数据网络之间网关的作用，能利用 IP 和不同的网络实现互通。同时，GGSN 可以与其他 UMTS 网络中的 GGSN 相互连接，实现移动终端在不同移动运营商所管辖的 UMTS 网络之间的漫游。GGSN 不仅可以提供地址转换业务，还可以作为防火墙，以保证所有的输入和输出数据都是经过鉴权的，从而增加移动核心网络的安全性。

(3) DNS 可以在 PDP 上下文激活过程中根据确定的 APN(access point name，接入点名)解析出 GGSN 的 IP 地址；还可以在 SGSN 间的路由区更新过程中，根据旧的路由区号码，解析出旧的 SGSN 的 IP 地址。

(4) BG(border gateway，边界网关)是用于两个运营商 GPRS 网络互联时的关口，可以实施一些安全/路由策略。

(5) CG(charging gateway，计费网关)可以采用独立的计费网关或采用嵌入 GSN 节点的计费网关，实施采集、存储并向计费中心传送话单。

(6) UMTS 网络提供数据承载通道,分为两段:移动台到 SGSN 基于 LLC(逻辑链路控制)协议的数据链路;SGSN 到 GGSN 之间基于 TCP/IP 协议和 GTP(隧道传输协议)的数据隧道。

(7) GTP 主要负责 SGSN 和 GGSN 之间分组路由管理和传输,是在 UMTS 网络内部支持分组数据终端的移动性专用协议,GTP 应用在 SGSN 和 GGSN 之间,为各个移动台建立 GTP 通道,GTP 通道可以供两个主机交换数据,SGSN 从 MS 接收数据包,并在 GTP 包头进行封装,然后才通过 GTP 通道转发到 GGSN,GGSN 在接收数据时先进行解封,然后再转发给外部主机。GTP 由处于两个 GSN 中有相互关系的 PDP(packet data protocol,分组数据协议)上下文定义。创建 GTP 隧道的过程就是激活这对 PDP 上下文的过程,涉及 GTP 创建的规程有三个:PDP 上下文激活规程、网络请求的 PDP 上下文激活规程以及跨 SGSN 路由区更新规程。

2. 传统隧道方案

1) 同一 SGSN 内的两 MS 通信(如 MS1 向 MS4 发包)

其信令流程如图 3-8 所示。

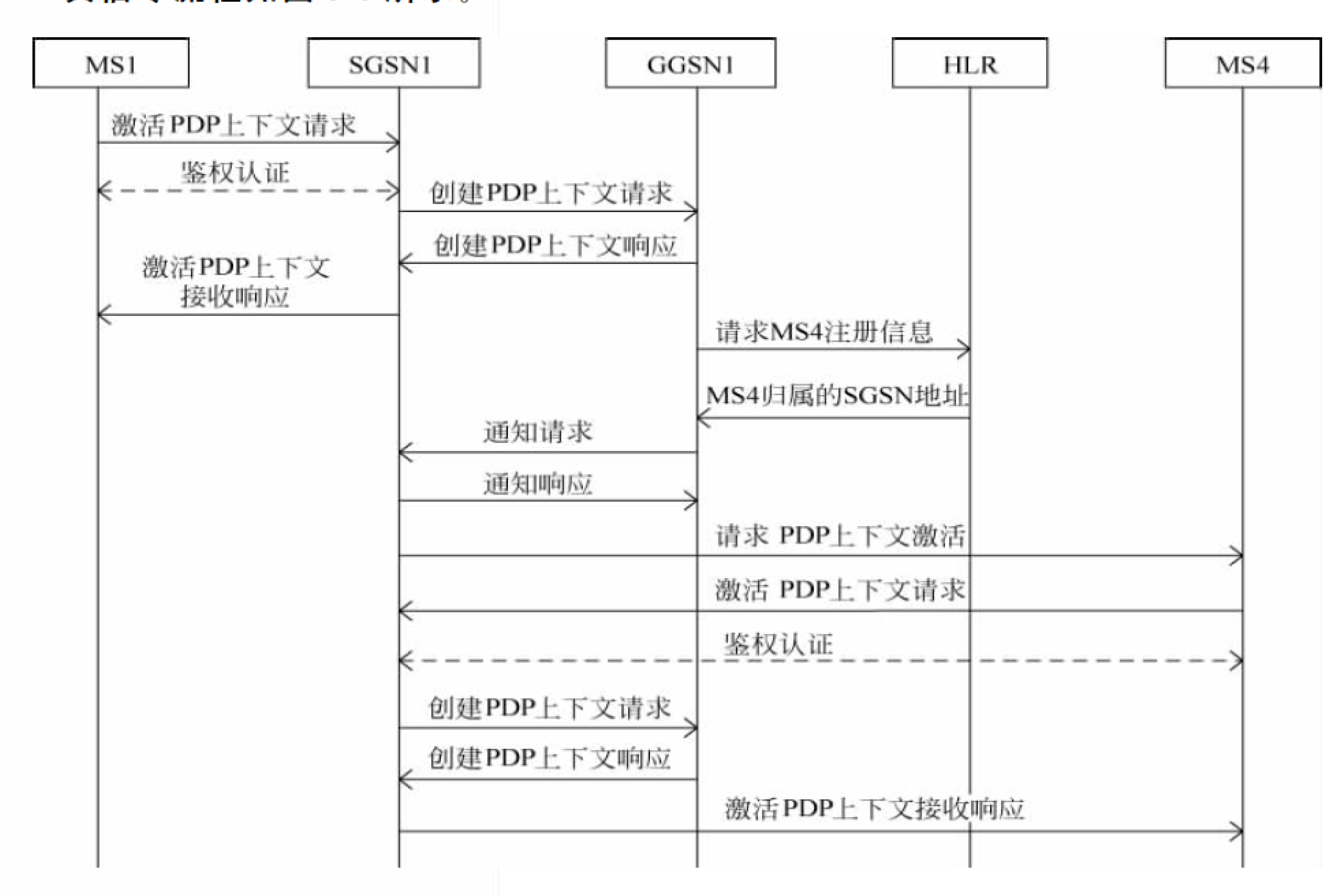


图 3-8 传统同一 SGSN 内的 MS 之间通信的信令流程

传输数据时,SGSN1 与 GGSN1 之间建立了隧道,数据传输路径为 MS1→BSS1→SGSN1→GGSN1→SGSN1→BSS2→MS4。

2) 不同 SGSN,同一 GGSN 的两 MS 通信(如 MS1 向 MS5 发包)

其信令流程如图 3-9 所示。

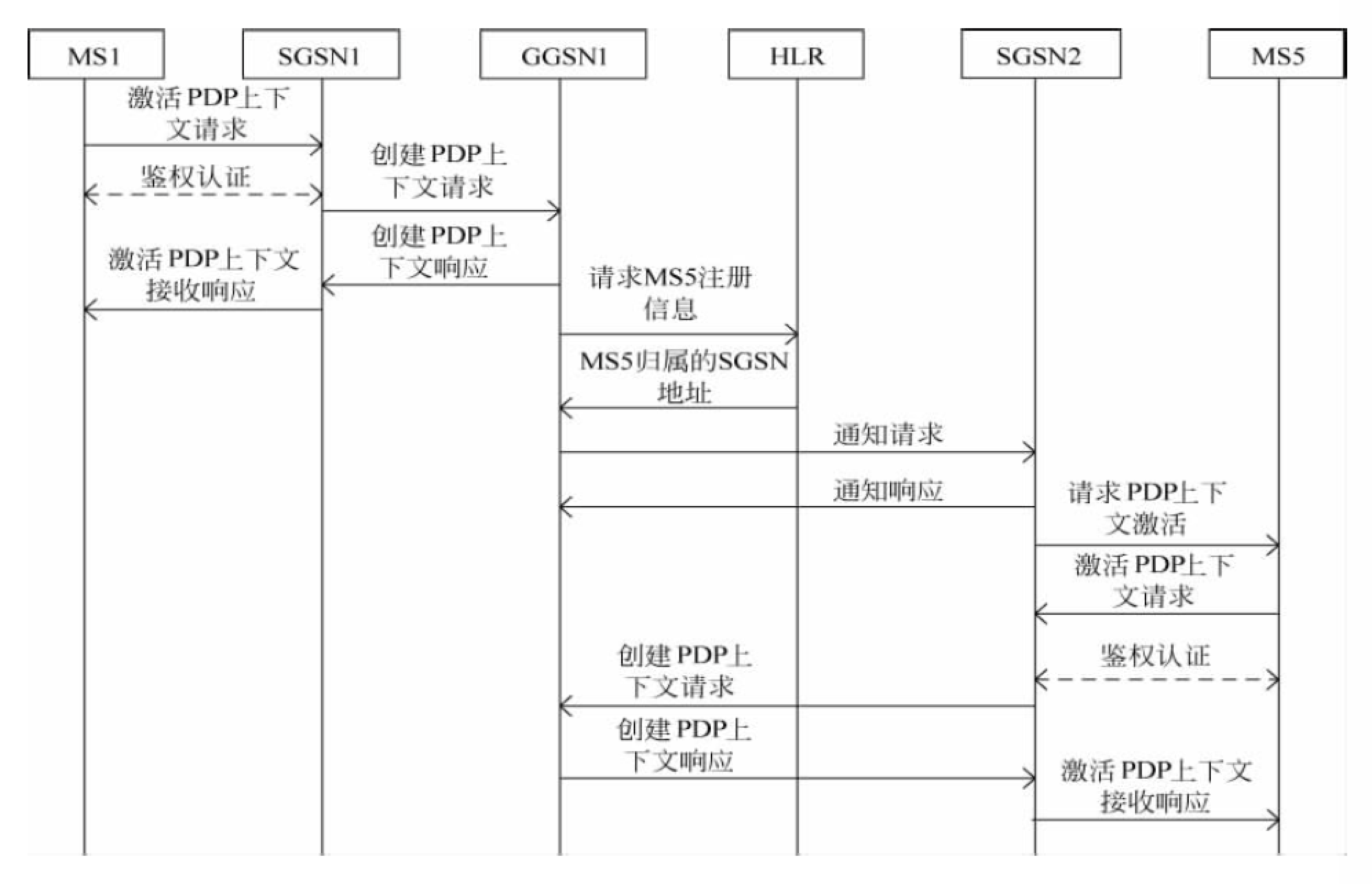


图 3-9 传统不同 SGSN 内的 MS 之间通信的信令流程

传输数据时，SGSN1 和 GGSN1、SGSN2 和 GGSN1 之间都要建立隧道，数据传输路径为：MS1→BSS1→SGSN1→GGSN1→SGSN2→BSS3→MS5。

3）MS 在 SGSN 之间切换（如 MS 由 SGSN1→SGSN2）

其信令切换流程及数据传输如图 3-10 所示。

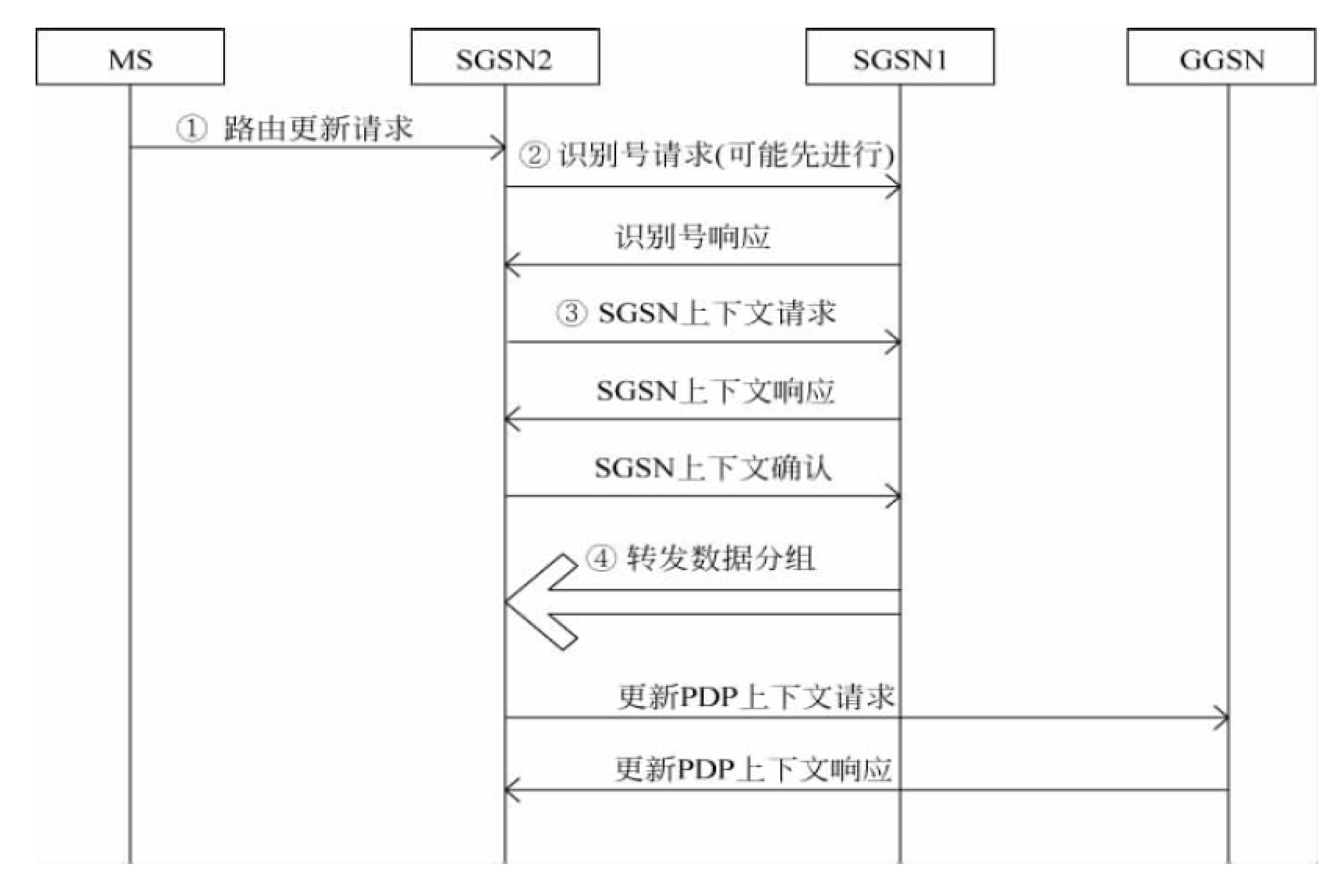


图 3-10 跨 SGSN 路由区更新

图 3-10 中，②中的识别号指的是 MS 的 IMSI，③中的 SGSN 上下文指 MS 的 MM (mobility management)上下文和 PDP 上下文，④中的转发数据是通过建立一个或多个隧道进行的。

为了传输数据，SGSN1 与 GGSN1、SGSN1 与 SGSN2、SGSN2 与 GGSN1 之间都要建立隧道。

3.1.4 异构网络的移动性解决方案

为了改善分层结构中存在的负载集中和单点故障的问题，本节将介绍基于分布式 MAP 结构的自适应位置管理方案，并为了进一步解决 HMIPv6 中的 MAP 单点故障问题，提出鲁棒移动性管理方案。

1. 自适应位置管理

为了改善基本移动 IP 的性能和可扩展问题，研究人员引入了分层结构的概念。但是，分层结构又会导致负载集中和单点故障的问题，因此本节将介绍一种基于分布式 MAP 结构的自适应位置管理方案。该方案通过采用自适应 MAP 选择算法来有效地进行 MAP 负载分担。

1）分布式 MAP 结构

基本 HMIP 方案的 MAP 是分层结构，上层 MAP 是信令和业务的集中点，容易造成节点负载过重，同时也引入了单点故障问题，降低了整个网络的鲁棒性。针对这个问题，有资料提出了全分布式的 MAP 结构，但是 MAP 的分布呈现一个“平坦”结构，不利于有效的管理[10]。本书重点介绍一种介于两者之间的分布式 MAP 结构，如图 3-11 所示。

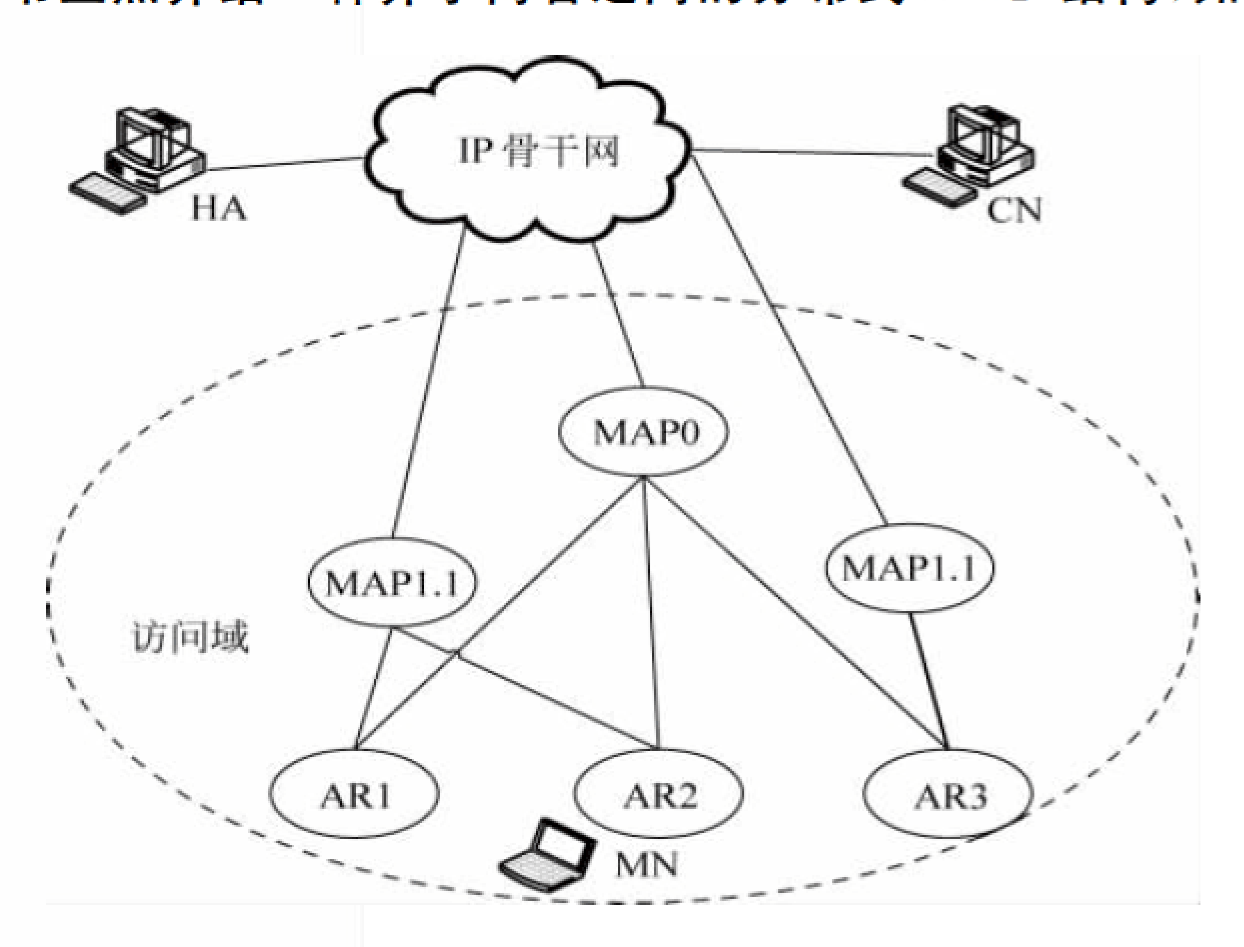


图 3-11　分布式 MAP 结构

在图 3-11 中，由一个或少数几个 MAP 覆盖整个访问域，如 MAP0。其余 MAP 覆盖区域较小的热点地区，如 MAP1.1 和 MAP1.2。MAP 域之间可以部分或全部重叠。一般来说，MAP0 与 AR 的距离要远一些。在这种分布式 MAP 结构中，首先 MAP 之间是

分布式关系，可以有效地分担各 MAP 的负载，不会将负载集中于高层 MAP。其次 MAP 和 AR 之间是分层结构，可以有效加强 MAP 的可管理性。分布式 MAP 结构也符合大范围覆盖和“热点”区域覆盖的原则，更利于部署。

2）自适应 MAP 选择算法

自适应 MAP 选择算法是根据 MN 的移动特性、会话达到率以及 MAP 的负载、优先级和距离等参数来动态地选择 MAP。与其他 MAP 选择算法相比，自适应 MAP 选择算法考虑了更多的综合因素，选择的 MAP 更为合理。而且自适应 MAP 选择算法中不需要事先设定阈值，具有更广泛的适用条件。算法具体操作过程如下：

（1）MN 收到 RA（路由器广播）的广播消息后，从 MAP 列表中去掉优先级为“不选”的 MAP 选项。

（2）计算出各 MAP 对应的系统开销值，并按计算结果对可选 MAP 进行降序排列。

（3）根据 MAP 排序结果和相应的优先级来确定最终 MAP 选择结果。假设 MAP 排序结果为 MAP_R1、MAP_R2、MAP_R3 等，那么判断的流程如图 3-12 所示。

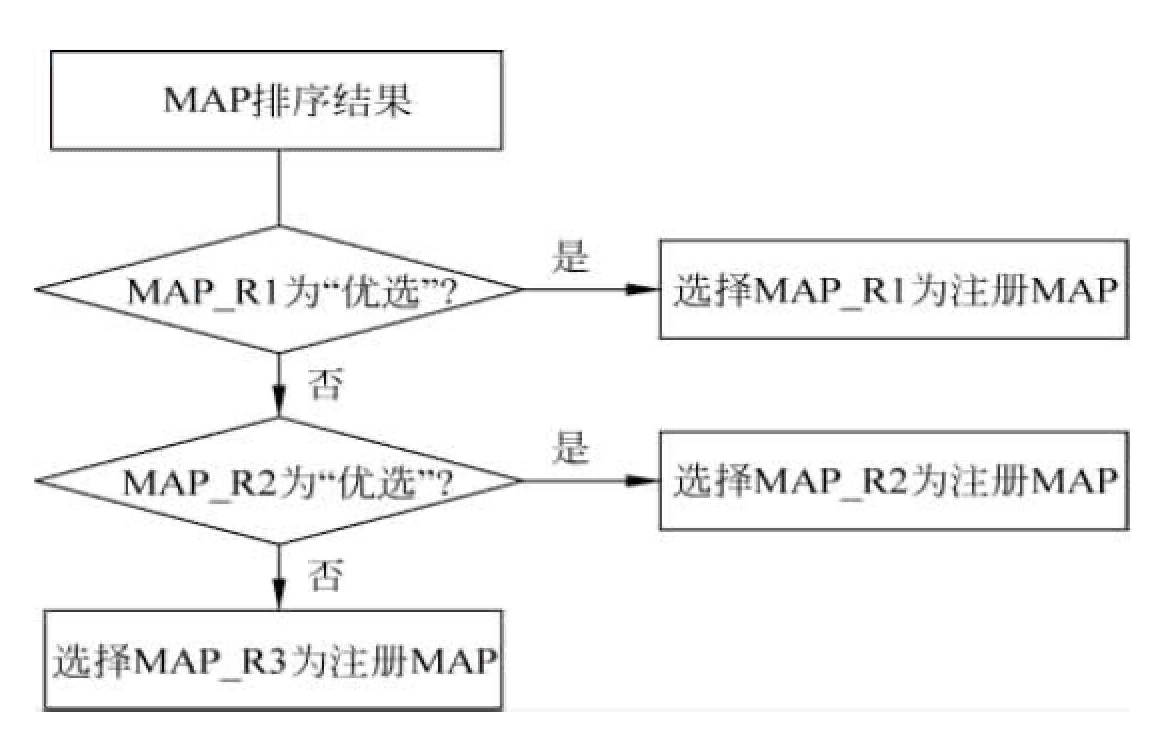


图 3-12　MAP 排序结果的判断流程

自适应 MAP 选择算法的步骤（2）中，把 MN 注册后产生的移动性管理系统开销为选择依据，系统开销公式为

$$C = C_{\text{HMIP-LU}} + C_{\text{HMIP-PD}} \tag{3-1}$$

式中，$C_{\text{HMIP-LU}}$是分层移动 IP 的位置开销；$C_{\text{HMIP-PD}}$是分层移动 IP 的分组传递开销。

$$C_{\text{HMIP-LU}} = n_{\text{inter}} \cdot (C_{\text{uh}} + n_{\text{CN}} C_{\text{uc}}) + n_{\text{intra}} \cdot C_{\text{ul}} \tag{3-2}$$

式中，n_{inter}是 MN 在单位时间内发生域间切换的次数；n_{intra}是 MN 在单位时间内发生域内切换的次数；C_{ul}是 MN 向 MAP 进行本地绑定更新的信令传输和处理开销。

$$C_{\text{HMIP-PD}} = C_{\text{PD}}^{\text{CN-HA}} + C_{\text{PD}}^{\text{HA-MAP}} + C_{\text{PD}}^{\text{MAP-AR}} + C_{\text{PD}}^{\text{AR-MN}} \tag{3-3}$$

如果采取了路由优化，CN 可以查找绑定缓存信息，直接向 MN 的区域转交地址（RCoA）发送分组数据，则分组传递开销公式为

$$C_{\text{HMIP-PD}} = C_{\text{PD}}^{\text{CN-MAP}} + C_{\text{PD}}^{\text{MAP-AR}} + C_{\text{PD}}^{\text{AR-MN}} \tag{3-4}$$

3）基于切换强度测量的分层结构动态调整算法

在分层结构的同一个管理域中，移动节点的位置改变并不会给上一级管理域带来信令开销，从而大大减少了切换时间和信令开销。但是由于网络业务和用户移动特性是不

断变化的，预先设计好的分层结构如果不能根据网络的实际情况做相应的调整，就会使得分层结构失效甚至导致更多的开销，从而引起网络性能下降。而基于切换强度测量的分层结构动态调整算法可以让切换强度最高的小区聚合至同一个管理域，从而使发生在管理域间的切换转化为同一管理域内的切换，提高了切换速度和节点流量。

在分层结构动态调整算法中，做如下定义：

（1）管理域 D——由多个移动管理代理组成的，对所属终端进行管理的区域。

（2）邻域 N——如果位于两个不同管理域中的两个移动管理代理之间只有一跳距离，则称这两个管理域为邻域 N。管理域 D_i 的所有邻域的集合记为 $N(i)$。

（3）切换强度 $H_{i,j}(\Delta t)$——一定时间间隔 Δt 内，两个邻域 D_i 和 D_j 之间发生切换的次数。

（4）紧邻域 $N_c(D)$——一定时间间隔内，某管理域 D 的邻域中切换强度最大的那个邻域。

则管理域 D_i 在 Δt 时间段内发生的切换次数 $H_i(\Delta t)$ 为

$$H_i(\Delta t) = \sum_{j \in N(i)} H_{i,j}(\Delta t) \tag{3-5}$$

管理域 D_i 和 D_j 之间的相对切换率 $HF_{i,j}(\Delta t)$ 为

$$HF_{i,j}(\Delta t) = \frac{H_{i,j}(\Delta t)}{H_i(\Delta t)} \tag{3-6}$$

构造管理域 D_i 与邻域之间相对切换率矩阵 $\boldsymbol{H}_i(t)$ 为

$$\boldsymbol{H}_i(t) = \begin{pmatrix} HF_{00} & HF_{01} & \cdots & HF_{0N} \\ HF_{10} & \cdots & & \\ \vdots & \vdots & & \vdots \\ HF_{N0} & 0 & \cdots & 0 \end{pmatrix} \tag{3-7}$$

将矩阵中邻域切换率大于 $1/N$ 的项对应的邻域设为“备选邻域”，并从中挑选邻域切换率最大项对应的邻域为“紧邻域”。则该管理域 D_i 将与“紧邻域”合并为同一个上级管理域，从而将这两个邻域间的切换转换为域内切换。这种管理域的自组织，可以提高移动节点的流量和切换速度，明显节省了信令开销。

2. 鲁棒移动性管理

自适应位置管理方案虽然可以在一定程度上减小 MAP 失效产生的影响，但并不能使受影响的 MN 从故障中尽快恢复。为了解决 HMIPv6 中的 MAP 单点故障问题，下面重点介绍鲁棒移动性管理方案（robust mobility management scheme，RMM 方案）。

鲁棒移动性方案分为两部分，即分别从终端侧和网络侧发现 MAP 故障并恢复。终端侧方案主要通过 MN 和 CN 的操作来发现和恢复 MAP 故障，使正在进行的通信不受影响，需要 MAP 部分的辅助。网络侧方案是基于冗余备份的，在访问域中单独设置一个可靠存储器，利用检查点策略进行绑定信息的存储和恢复，两部分既可单独使用，也可联合使用，从而进一步提高系统的容错性能。

1）基于终端侧的 MAP 故障发现和恢复机制

具体方案的实施过程如图 3-13 所示。

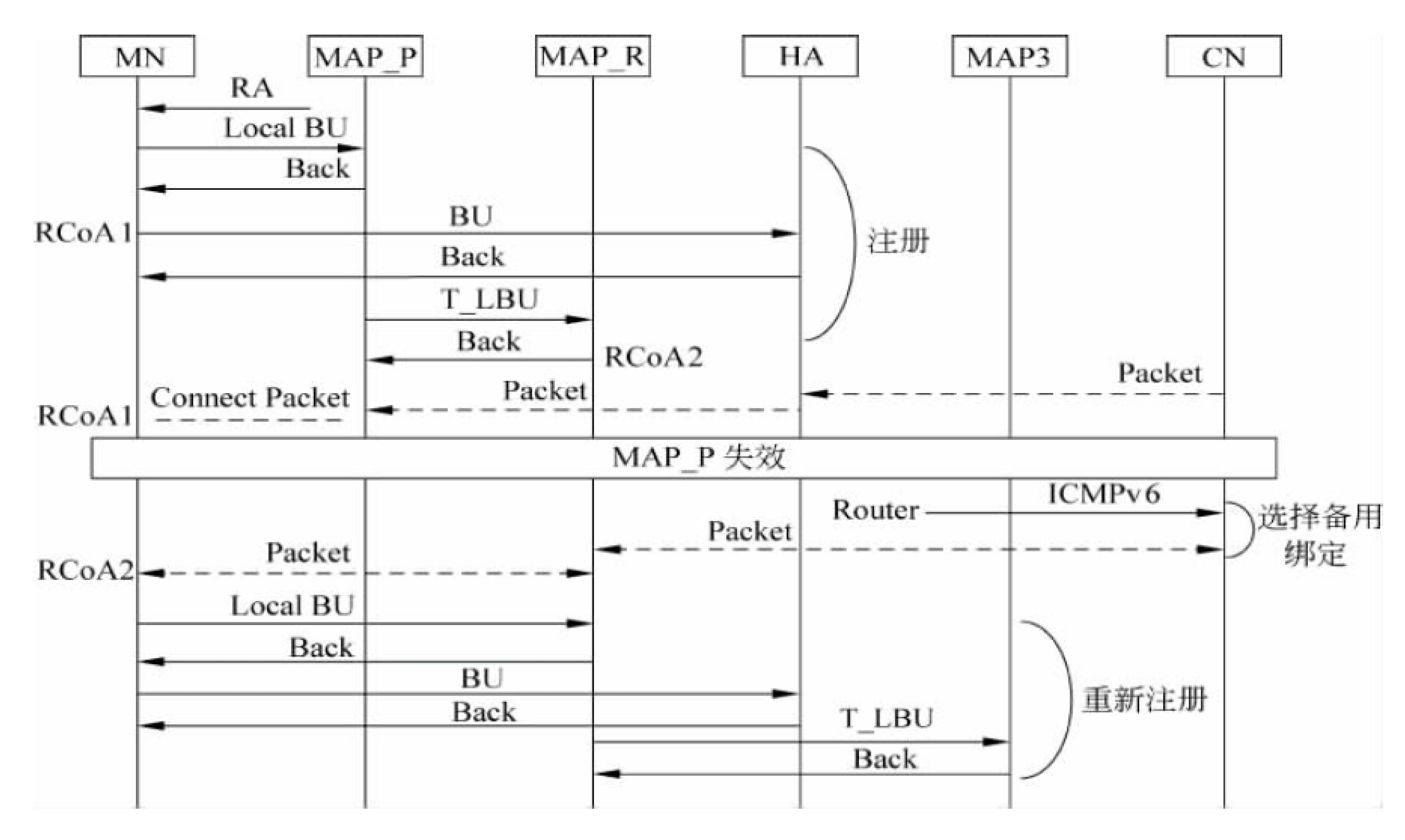


图 3-13 采用终端侧方案的 MN 注册、MAP 故障发现和恢复流程

(1) MN 的注册过程。MN 在进入一个新的访问域时，如果它收到两个或多个 MAP 的代理广播消息，会按照自适应 MAP 选择算法选择两个 MAP，最合适的 MAP 作为主用 MAP(MAP_P)，次合适的 MAP 作为备用 MAP(MAP_R)，并配置主用 RCoA 和备用 RCoA。MN 向主用 MAP_P 进行注册时，绑定更新消息中包含两个 RCoA 的信息，MAP_P 在收到绑定更新后，仍然按照正常的处理流程对 MN 的绑定更新进行应答和存储。

MAP_P 根据绑定更新中备用 RCoA 的前缀信息与其他 MAP 的前缀进行比较，判断备用 MAP 为 MAP_R。然后向 MAP_R 转发绑定更新消息。MAP_R 收到转发的绑定更新消息后，若判断是备用的绑定更新，就会将 MN 的转交地址保存在备用绑定缓存中，并向 MAP_P 发送应答。如果 MAP_P 在规定时间内未收到 MAP_R 的绑定确认消息，则进行重发。由于 MAP 之间是有线线路，所以这种重发的情况比较少。注意，MAP_P 是首先向 MN 发送绑定确认，而不需要等待 MAP_R 的应答。

MN 在向 HA 和 CN 发送绑定更新消息时，也是发送包含两个 RCoA 信息的绑定更新消息。HA 和 CN 可以从本地绑定更新消息中的家乡地址(home address)字段中得到主用 RCoA，然后查找标识位 P，如果 P 为 1，就可以从 MAP 选项(MAP option)中得到备用 RCoA，并将主、备用 RCoA 存储在相应的主、备用绑定缓存中。

在此 MAP、MN、CN 和 HA 上都保存了 MN 的主、备用绑定信息(RCoA_P 和 RCoA_R)。需要注意的是，MN 的主、备用绑定信息在 MN、CN 和 HA 上是同时保存的，而在 MAP 上是分别保存的。例如，某一个 MN 的主、备用 RCoA 分别为 RCoA1 和 RCoA2，那么，该 MN 和与它通信的 CN 上都存储了 RCoA1 和 RCoA2，而 MN 在向 MAP 进行绑定更新时，RCoA1 存储在 MAP_P 的主用绑定缓存中，而 RCoA2 存储在 MAP_R 的备用绑定缓存中。所以，对于不同的 MN，即使它所选择的主用 MAP 一样，但备用 MAP 很可能不一样。

（2）故障发现。终端侧方案发现 MAP 故障的途径，主要是通过采用 ICMPv6 协议。正在通信的网络实体可以很快地发现节点故障，MAP 故障发现时间在毫秒级。

① MN 发送数据的情况。在采用 HMIPv6 后，MN 发送数据时多采用反向隧道向 CN 发送数据，分组数据经过一系列路由器被转发至正在服务的 MAP。如果服务 MAP 此时发生故障，则 MAP 之前的路由器由于不能将分组数据送达 MAP，将产生一个 ICMPv6 目的不可达消息，消息的目的地址是分组数据的源地址。MN 收到这个 ICMPv6 目的不可达消息，就可以发现服务 MAP 故障。

② CN 发送数据的情况。CN 向 MN 发送数据分两种情况。第一种情况：CN 在 MAP 故障前就已经和 MN 进行通信，则 CN 上有 MN 的绑定缓存，其分组数据一定会先转发至 MN 的服务 MAP。如果 MAP 失效，CN 发送的数据包无法到达 MAP，就会收到由中间路由器返回的 ICMPv6 目的不可达错误消息，CN 即发现服务 MAP 的故障。第二种情况：CN 在 MAP 故障前没有与 MN 进行过通信，即 CN 上没有 MN 的绑定缓存，那么 CN 发往 MN 的分组数据一定会发至 MN 的 HA，再由 HA 隧道至 MN 的服务 MAP。如果 MAP 失效，HA 就会收到一个 ICMPv6 目的不可达错误消息，并据此判断 MAP 发生了故障。

（3）故障恢复。本方案中的故障恢复主要是由 MN 和 CN 来操作的，分别叙述如下。

① MN 发现 MAP 故障的情况。

如图 3-13 所示，当 MN 通过 ICMPv6 消息或代理广播消息检测到 MAP_P 的故障，将立即采用备用 RCoA 作为主用 RCoA 来发送数据或信令。同时，MN 重新选定备用 MAP，向 MAP_R、HA 和 CN 发送绑定更新请求，这时，MAP_R 为 MN 的主用 MAP。CN 若收到源地址为备用 RCoA 的分组数据，CN 可以判断出 MAP_P 发生了故障，将 MN 的备用 RCoA 替代主用绑定缓存中的主用 RCoA。

② CN 发现 MAP 故障的情况。

当 CN 通过 ICMPv6 消息检测到 MAP_P 的故障，将立即采用 MN 的备用 RCoA 替代主用绑定缓存中的主用 RCoA，并用该绑定向 MN 发送数据。这时，MN 也将收到由 MAP_R 转发来的数据，MN 就判断出 MAP_P 发生了故障。MN 将 MN 的备用 RCoA 替代主用绑定缓存中的主用 RCoA，并重新选定备用 MAP，向 MAP_R、HA 和 CN 发送绑定更新请求。

③ HA 发现故障的情况。类似 CN 发现故障的情况，当 HA 通过 ICMPv6 消息检测到 MAP_P 的故障，将立即采用 MN 的备用 RCoA 替代主用绑定缓存中的主用 RCoA，并将 CN 的数据重新隧道至 MAP_R。这时，MN 也将收到由 MAP_R 转发来的数据，MN 就判断出 MAP_P 发生了故障。MN 将 MN 的备用 RCoA 替代主用绑定缓存中的主用 RCoA，并重新选定备用 MAP，向 MAP_R、HA 和 CN 发送绑定更新请求。

基于终端侧分层移动 IP MAP 故障发现和恢复机制的优点是 MAP 可靠性高，非故障开销较小，具有负载分担的特点，故障发现和恢复的时间很短，在故障恢复的过程中不会引起某一个 MAP 性能的突然下降。但该方案对 HMIPv6 协议做了少部分的扩展，并且 MN、CN、HA 和 MAP 都需要对主、备用绑定缓存进行处理和存储，对终端不透明。

2）基于网络侧的 MAP 故障发现和恢复机制

终端侧方案主要的缺点是对 MN 和 CN 不透明，需要修改协议信令。另外，所有受

影响的 MN 在检测到 MAP 故障后，仍需要进行重新注册，过程与 HMIPv6 协议中的操作一样，即故障恢复开销还是很大的。

针对终端侧方案的不足，这里介绍另一种鲁棒移动性管理方案——基于网络侧的分层移动 IP MAP 故障发现和恢复方案。

此研究基于以下假设：

（1）基于网络侧的方案采用 ICMPv6 协议进行 MAP 故障发现，并且需要 MAP 和可靠存储器都支持 ICMPv6 协议。

（2）可靠存储器，即不会发生故障的存储器。其中存储的绑定缓存项类似 MAP 中存储的绑定缓存项，是有生命周期的，对于超过生命周期而没有进行更新的绑定项，需要进行删除。

（3）MAP 与可靠存储器之间要建立安全联合，保证数据传送的安全，MAP 与可靠存储器之间采用 TCP 传递消息。

方案中增加了三条控制消息：绑定缓存更新消息（SBU-update）、绑定缓存下载消息（SBU-download）和绑定缓存确认消息（SBU-ack）。这三个消息均由 ICMPv6 消息扩展得到，绑定缓存更新消息用于 MAP 向可靠存储器定期发送绑定缓存信息；而绑定缓存下载消息和绑定缓存确认消息用于可靠存储器向 MAP 下载绑定缓存信息。

具体方案描述如下：

网络侧方案是基于冗余备份，在访问域中单独设置一个可靠存储器，利用检查点策略进行绑定信息的存储和恢复。可靠存储器的主要作用是存储绑定缓存的项目，并不承担类似备份 MAP 的功能，因为备份 MAP 的功能除备份绑定缓存的功能之外，还需具备转发数据包等其他功能，如果可靠存储器承担 MAP 的功能，则处理能力和安全性势必受到影响，这与假设为绝对安全的存储器存在矛盾。

网络侧方案网络结构如图 3-14 所示，可靠存储器与所有 MAP 之间均存在安全联合（SA）。当访问域中需要加入新的 MAP 时，新 MAP 也只需要和可靠存储器建立安全联

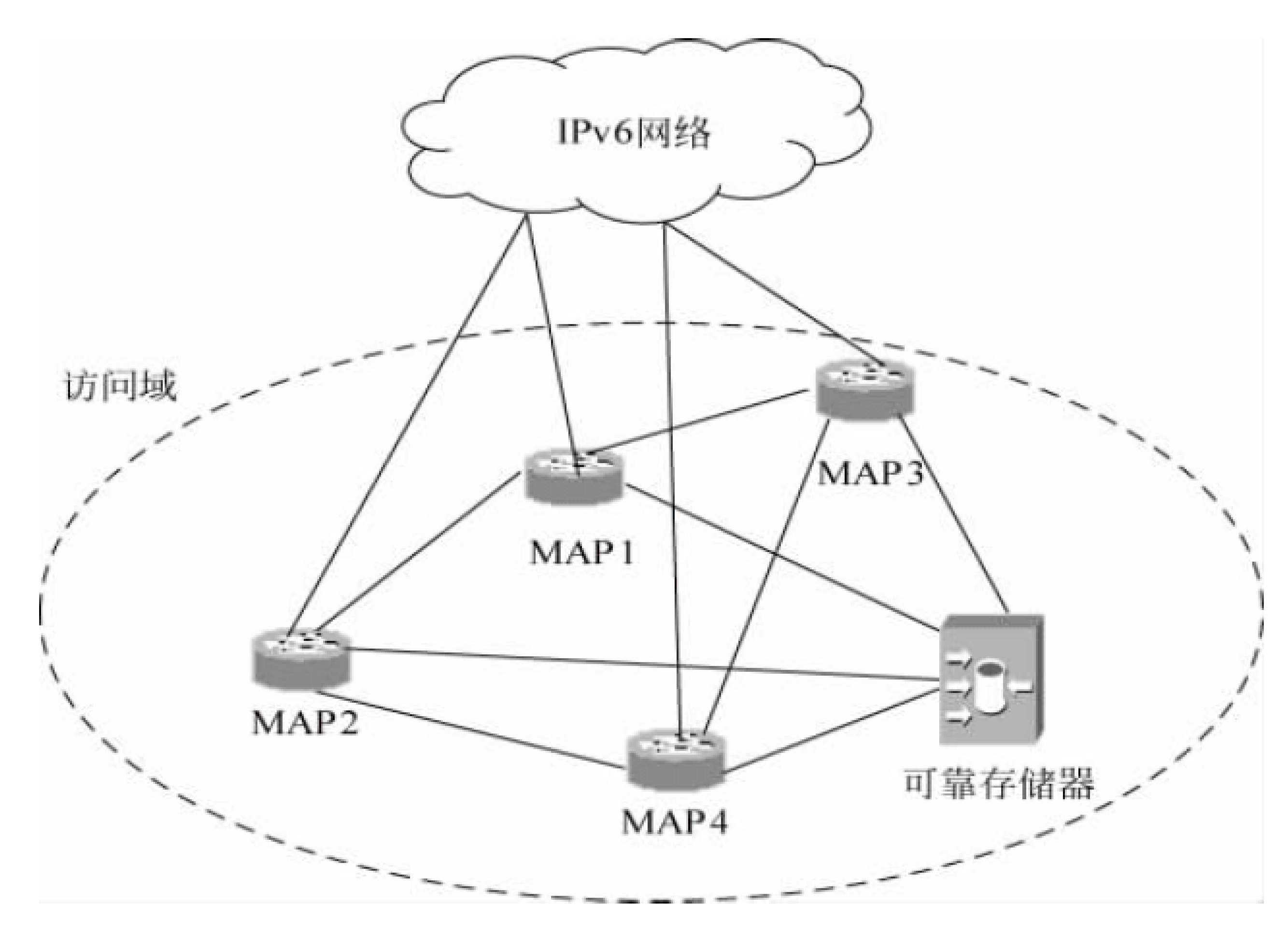


图 3-14　网络侧方案网络结构图

合。可靠存储器中存储了访问域中所有 MAP 的信息，包括地址信息和绑定缓存信息。MAP 的地址信息一般不改变，可以手工配置到可靠存储器中。绑定缓存信息是不断更新的，需要 MAP 定期向可靠存储器更新相应的绑定缓存信息。

在网络侧方案中，MN、CN 和 HA 的操作与 HMIPv6 协议完全一致。只有 MAP 的操作略有不同，如图 3-15 所示。

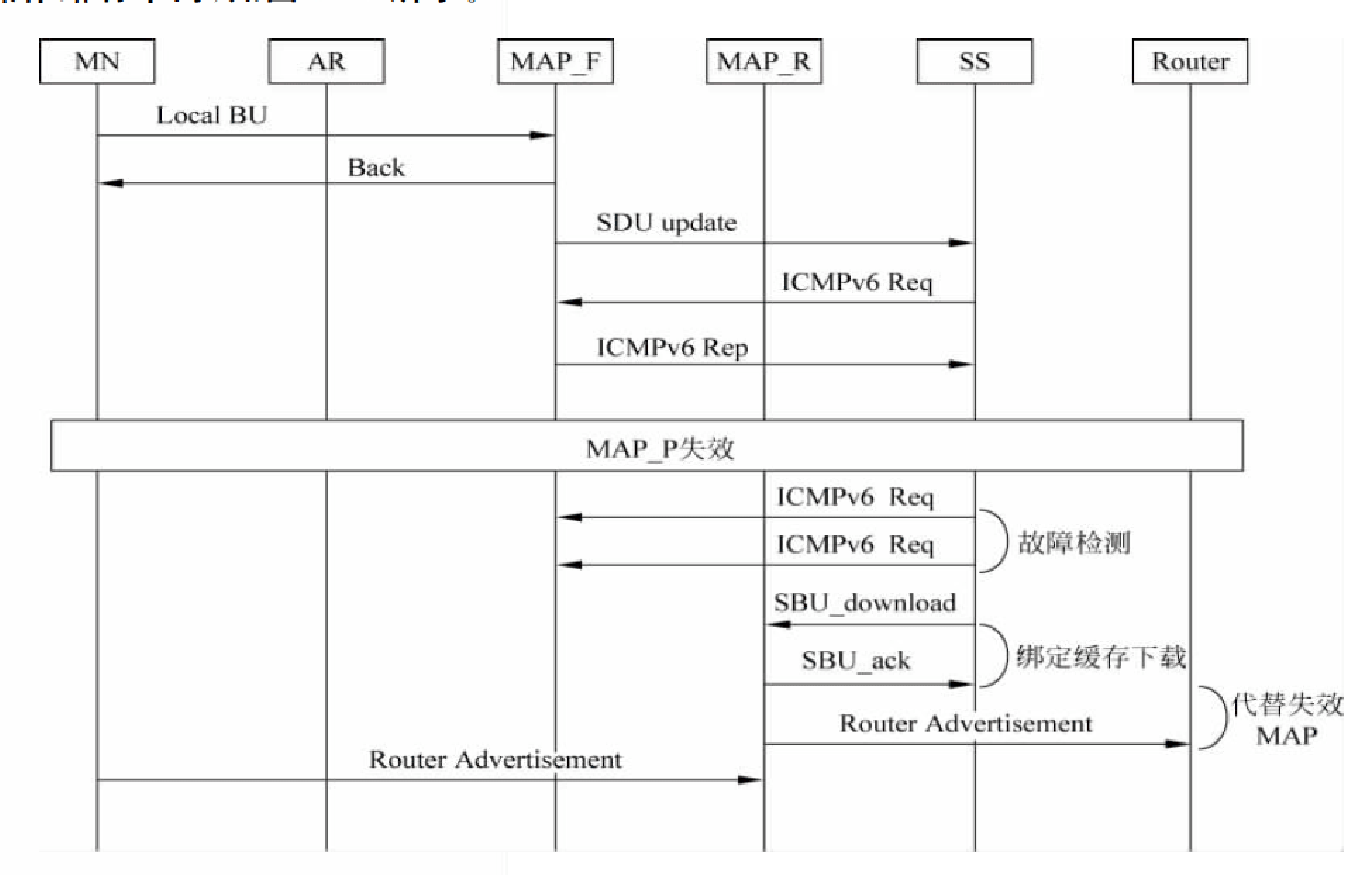


图 3-15　基于网络侧方案的 MAP 故障发现和恢复流程

MAP 操作过程为：当 MN 向 MAP 发送绑定更新消息时，MAP 除了保存 MN 的绑定信息以及发送绑定应答以外，还定期将 MN 的绑定信息发送到可靠存储器上。如果 MAP 每收到一条绑定更新消息，就向可靠存储器发送，则 MAP 对可靠存储器的操作比较频繁，交互的信令也比较多。通常情况下，MAP 周期性地发送信息。但是发送周期的选取也需要一个合适的尺度。周期短，则交互仍然频繁，但可靠存储器备份 MAP 的绑定缓存信息比较及时。周期长，则减少了交互次数，但绑定缓存信息备份不够及时，当 MAP 发生故障时，可能导致某些绑定缓存的丢失。为了避免 MAP 频繁地操作可靠存储器，同时又提高备份信息的及时性，在 MAP 上采取动态周期存储算法（dynamic periodic saving algorithm，DPSA）来发送绑定更新信息。动态周期发送算法的思想比较简单，选取较短的周期（即存储定时器的值较小）发送绑定缓存信息，以保证绑定缓存信息能够及时地存储到可靠存储器中。当 MAP 在一个周期内收到新的绑定更新消息，则在存储定时器到期时，将所有新的绑定更新发送到可靠存储器。如果 MAP 在一个周期内没有收到新的绑定更新消息，存储定时器就暂停。若再次有新的绑定更新到来，则 MAP 立即将该绑定更新发送到可靠存储器，并重新启动存储定时器。该算法的优点是在没有发送需要的情况下，减少了 MAP 发送绑定更新的次数。

(1) 可靠存储器的操作。可靠存储器在收到 MAP 周期发送的绑定缓存信息后,将绑定缓存信息存储在相应 MAP 的存储空间。同时,每个绑定缓存都有生命周期,对于超过生命周期而没有进行更新的绑定项,可靠存储器需要进行删除。当可靠存储器发现某个 MAP 失效后,将选择一个备用 MAP 进行替代,并在可靠存储器中将故障 MAP 的所有绑定缓存信息传递下载到备用 MAP 中。

(2) 故障发现。可靠存储器通过定期发送 ICMPv6 回声请求和回声应答消息来判断是否有 MAP 发生故障。其故障发现时间根据发送 ICMPv6 消息的周期来确定,为了尽量缩短 MAP 故障发现时间,这里定义 ICMPv6 消息的发送周期为 1s,如可靠存储器未收到相应的回声应答消息,则缩短周期为 0.5s,最大重发次数为三次。

(3) 故障恢复。当访问域有 MAP 故障(MAP_F)时,可靠存储器 SS 将启动替代 MAP 选择算法,选择唯一的一个 MAP(MAP_R)进行替代操作。为了平衡各个 MAP 的负载,可尽量选择负载小的 MAP 作为替代 MAP。通过 SBU_req 和 SBU_rep 消息,MAP_R 从可靠存储器中获得 MAP_F 的绑定信息表,即获得了 MAP_F 管辖区域内正在服务的 MN 列表。MAP_R 执行代理 ARP,将 MAP_F 的 IP 地址和 MAP_R 的硬件地址进行绑定。而且 MAP_R 必须对发向 MAP_F 的 ARP 请求进行应答。所有发往 MAP_F 的分组都将由 MAP_R 截取。MAP_R 要代替 MAP_F 对 MN 的注册请求进行应答,并接受 MN 的注册更新消息,还要根据下载的绑定信息将分组传送至对应的 MN。MAP_R 完成这些操作后,就完全替代了 MAP_F。并且这些操作对于 MN 是透明的,即 MN 不会察觉 MAP_F 发生了故障,也不需要进行额外的操作。

由以上讨论可知,基于网络侧的分层移动 IP MAP 故障发现和恢复方案非故障开销小,对 MN/CN 完全透明。尤其适合 MAP 故障时未进行通信的节点,这部分节点不能发现 MAP 故障,也不需要重新注册,简化了通信节点的操作并节省了无线资源。网络侧方案是基于冗余备份,主要缺点是故障发现和恢复的时间较长。

3.2 异构无线网络资源管理

传统无线资源管理都是针对单一网络内部的资源进行管理,没有考虑其他网络的可用资源,其目标是在有限带宽的条件下,为该网络内的无线用户终端提供业务质量保障。但是这种孤立的资源管理方式,不能有效地利用未来多无线接入的网络能力,无法协调各无线网络之间的资源,无法根据用户的业务特性实时地分配最佳网络资源,服务质量也无法保证,网络的资源利用率低。

在下一代异构无线网络环境中,属于不同网络运营商和提供商的各种不同空中接口、覆盖范围、接入速率、容量、移动性和业务特性的无线接入网络将同时存在并协同工作,对各种网络独立运营和维护不是有效的管理方法。为此,需要引入新的无线资源管理方案,从整体的角度对各种 RAT(radio access technology,无线接入技术)提供的全部无线资源进行统一的管理。

对于无线资源管理的模型,目前主要有三种,即公共无线资源管理(CRRM)模型、多无线资源管理(MRRM)模型以及联合无线资源管理(JRRM)模型。

围绕无线资源管理问题,网络选择、垂直切换、接纳控制是重点研究的方向,网络选

择将在3.3节讨论，本节将围绕无线资源管理模型、接纳控制机制进行研究。

3.2.1 资源管理模型

1. 公共无线资源管理(CRRM)模型

3GPP在制定规范的时候就已经考虑了多种无线接入技术共存的融合网络场景，提出了通用无线资源管理(CRRM)的概念，为各种业务选择最合适的无线承载，以增强网络的QoS管理能力。

CRRM作为融合网络中接入管理者，其作用就是在切换和呼叫建立过程中，对不同的候选目的小区进行优先处理。CRRM的主要优势包括：通过负载均衡，从而降低阻塞率和提高系统资源利用率；为各种业务选择最合适的无线承载，以增强网络的QoS管理能力。

然而，CRRM也存在一定的局限性，这主要体现在两方面：首先，无论呼叫建立还是系统间切换，接入网络选择主要考虑了负载因素，而没有考虑其他众多可以影响选择的因素，如用户偏好、移动速度和趋势、当前网络信号状况以及QoS等；其次，CRRM仅针对UTRAN、GERAN等蜂窝网络，而不包括其他类型的网络，如无线局域网WLAN、无线个域网WPAN等。

整个异构无线资源管理的关键问题就是对无线资源池进行有效管理以确保无缝链接，因此在异构无线网络资源管理中分两个级别，接入网内部的资源管理RRM和公共无线资源管理CRRM，其中RRM负责管理一个网络的本地无线资源池。CRRM负责管理整个异构网络间的资源池，它基于集中式的管理结构，对由不同RRM实体所控制的资源池进行统一的协调管理。

RRM与CRRM之间的交互分为两个主要功能[11]，第一个是信息报告功能，RRM实体向控制它的CRRM实体报告小区的情况，报告可以是周期性的，也可以是触发式的，报告内容包括小区的静态信息(如小区能力、容量、小区间位置关系等)和小区的动态信息(如小区负载、总带宽、干扰等)。

第二个功能是RRM决策支持功能，指的是在CRRM与RRM交互时进行决策的方式，即在CRRM的控制下，影响RRM进行决策的方法，可以有两种决策方案，一种是RRM实体是决策的主管，以CRRM实体的决策为辅，这种交互通常所需的时间较长。第二种是CRRM实体做中心决策，以RRM的决策为辅，这种情况可以通过将局域RRM功能转变为CRRM功能来实现，这种情况下CRRM做决策的时间较短，仅在毫秒级别。现有文献根据RRM与CRRM之间的交互级别可以分为4种情况：低级、中级、高级和超高级，级别越低，所需要的决策时间也越长，其中低级交互所需时间的数量级为小时，而中级交互所需时间数量级为分钟，高级交互所需时间仅为毫秒级别。

值得一提的是，RRM和CRRM的交互程度与无线接入网络的耦合结构并不直接相关。后者主要指不同网络之间如何相互连接以及它们之间的相互作用的程度；前者则仅仅指无线资源管理过程中的操作。不过，不同的耦合结构导致实体通信过程中的延时不同，CRRM与RRM最高程度的交互只有在紧密和非常紧密的耦合结构下才更加灵活。

2. 多无线资源管理(MRRM)模型

针对下一代移动通信网中 Multi-Radio 应用场景,设计了一种通用的超 3G(B3G) Multi-Radio 接入架构 gMRA。采用了模块化方法,在架构模型中设置了两个关键的功能实体:Multi-Radio 无线资源管理(MRRM 模块)和通用链路层(GLL 模块)。两个功能实体之间相互协作,一方面对 B3G 中无线资源进行有效管理,实现异构无线系统间无缝切换。另一方面根据网络负载情况,合理调整业务分布,充分利用有限的带宽资源,完成接入网络间动态的负载均衡,提高全网的有效吞吐量。

在 gMRA 的体系架构中,MRRM 主要功能包括系统整体的资源管理、RRM 功能的补充、动态负载分配、准入控制、通用链路层控制、拥塞控制等。MRRM 作为控制平面的功能实体,根据业务承载要求和底层无线信道质量的变化,动态管理来自上层用户的数据流,在无线接入网之间实现合理调度。

MRRM 还控制不同 RAT 之间的切换,有效地调配所有无线资源,尽量满足应用层对 QoS 的需求。MRRM 的功能可分布在用户终端侧,也可分布在网络侧。由于下一代网络多种网络重叠覆盖的特点,MRRM 同时分布在网络侧和终端侧为最佳选择。

通用链路层(GLL)中一个重要的功能就是将来自高层的数据流动态倒换到合适的 RAT 中,而上层完成最佳接入路径的选择,通过控制正在服务的无线资源管理模块,对 GLL 进行配置和重新配置,根据无线资源的可用情况和来自底层的判决信息,使数据流在不同的 RAT 之间进行倒换[12]。

3. 联合无线资源管理(JRRM)模型

欧盟框架计划中曾对异构蜂窝网络中的无线资源管理工作进行研究,主要任务是设计和评估异构蜂窝网络中公共无线资源管理的具体策略和算法,提高 B3G 系统的无线资源利用率,其定义了联合无线资源管理(JRRM)模型,通过在不同的无线接入网络上设置一个集中的联合控制实体,从而进行联合的接纳控制、资源调度和负载控制。JRRM 设计的目标是保证用户的 QoS 需求,最大化网络的性能、容量与收益。

JRRM 提出了业务分流的思想,将业务划分为基本部分和加强部分,利用多个 RAT 的网络资源传输,在接收终端进行业务合并。其中,基本部分被认为是再现这种业务所必需的,所以只能承载在具有大范围覆盖的网络(如 UMTS),而增强部分则认为是用来提高用户的 QoS,一般承载在高速率的网络载体上。未来的 JRRM 模式不再局限于单一的集中式管理,而是可以采用集中式、分布式以及介于两者之间的分级式的管理方式。

JRRM 另一个特点就是需要终端乃至网络都具有可重配置性,如在各接入网络都具有内在的 QoS 保证机制,即能够在链路层实现一定的 QoS 保证,从而能够满足准入控制和联合资源调度的综合管理需求。

多接入选择(MRAS)作为 JRRM 中的关键技术,通过动态管理终端接入一个或多个不同的无线网络,可有效利用多接入增益。由多接入选择所带来的多接入增益包括两个方面:多接入分集和多接入合并。另外,JRRM 通过负载均衡以及动态频谱分配等技术[13],使得在多个可用无线网络之间能够以一种协调的方式自适应分配资源。

3.2.2 接纳控制

接纳控制(call admission control,CAC)决定了一个新的会话(连接)能否建立,当用户的服务请求到达时,接纳控制采用特定控制算法,综合分析当前的网络状态(包括已接纳的业务流信息、新进业务流的资源需求),判定当前网络的可用资源是否满足新进业务流的 QoS 需求,并做出接纳决定。接纳控制是实现网络 QoS 保障的重要手段,也是实现资源利用率与承诺 QoS 之间的折中平衡的关键机制,长期以来一直受到广大研究者的关注。

传统的单个网络的接纳控制就是在保证现有连接的 QoS 基础上判决是否接纳新的连接。相比于单一网络的接纳控制,将异构网络的接纳控制方案称为联合接纳控制机制(JCAC)。联合接纳控制机制所要解决的问题不仅仅是判定是否接纳新的连接,还包括为新连接判定接入哪个网络的问题。甚至在未来终端许可的情况下,接纳控制可能会允许终端同时与多个无线网络建立连接,传递同一个业务,从而获得比单个网络连接更好的服务质量。

接纳控制的分类很多,以采用的算法作为分类标准,可分为基于参数的接纳控制、基于测量的接纳控制以及基于策略的接纳控制;按照决策方式的不同,接纳控制方法也可分为分布式和集中式两类,其中分布式接纳控制方法又分为基于参数的接纳控制方法和基于测量的接纳控制方法。

然而目前对于异构网络下的 JCAC 的研究不算很多,本小节将重点分析三种联合接纳机制:基于模糊逻辑的联合接纳控制机制[14];基于负载均衡的联合接纳控制机制[15]和基于阈值的联合接纳控制机制[16,17],并进行了马尔可夫建模。

1. 基于模糊逻辑的联合接纳控制(FB-JCAC)

基于模糊逻辑的 JCAC 分为三个步骤:小区选择、本地接纳控制判决和 RAT 选择,如图 3-16 所示。

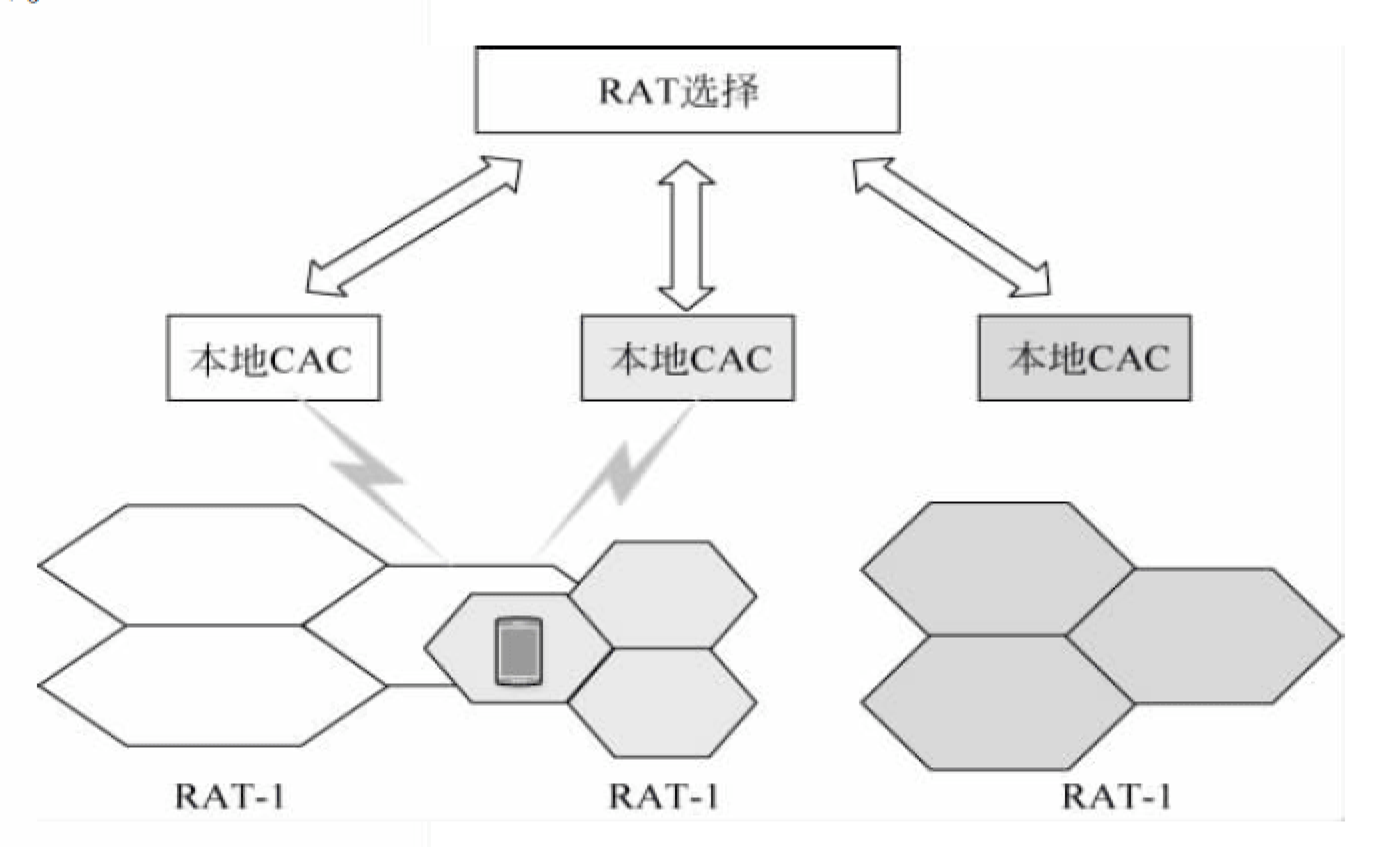


图 3-16 基于模糊逻辑的联合接纳控制(FB-JCAC)

当 MN 处于多个 RAT 重叠覆盖区域时，若有一个新的呼叫请求到达时，执行 FB-JCAC。小区选择阶段的主要判决条件是信号强度。在每种 RAT 中，移动终端选择信号强度最高的一个小区。因此，每种可选 RAT 对应一个最优小区。

之后由本地接纳控制算法决定引入的呼叫能否被接纳进入所选的小区。本地 CAC 使用的模糊逻辑控制器如图 3-17 所示，它由模糊器、干扰引擎、模糊规则库和解模糊器组成。

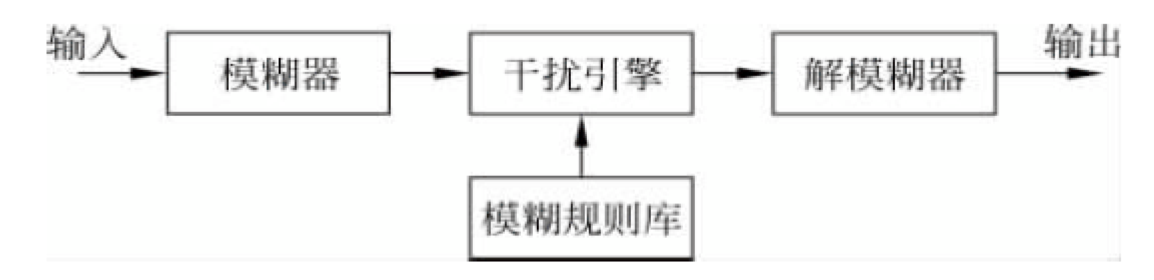


图 3-17　模糊逻辑控制器

模糊器将输入的测量值翻译成相应的模糊集合的语句值。以信号强度(SS)和网络负载(L)作为模糊变量，输出的语句参数为本地接纳判决(LAD)，因此 SS、L 和 LAD 的条件值定义如下：

$$\begin{aligned} &T(\mathrm{SS}) = \{\mathrm{Low}, \mathrm{Medium}, \mathrm{High}\} = \{\mathrm{L}, \mathrm{M}, \mathrm{H}\} \\ &T(\mathrm{L}) = \{\mathrm{VeryLow}, \mathrm{Low}, \mathrm{High}, \mathrm{VeryHigh}\} = \{\mathrm{VL}, \mathrm{L}, \mathrm{H}, \mathrm{VH}\} \\ &T\{\mathrm{LAD}\} = \{\mathrm{Reject}, \mathrm{WeakAccept}, \mathrm{Accept}\} = \{\mathrm{R}, \mathrm{WA}, \mathrm{A}\} \end{aligned} \tag{3-8}$$

干扰引擎使用预先定义的模糊规则针对每个 RAT，判决新连接能够被接纳到选择小区。预定义的规则如表 3-2 所示。

表 3-2　本地 CAC 的模糊规则

规　则	负　载	信号强度	判　决
0	VL	L	R
1	VL	M	A
2	VL	H	A
3	L	L	R
4	L	M	A
5	L	H	A
6	H	L	R
7	H	M	WA
8	H	H	WA
9	VH	L	R
10	VH	M	R
11	VH	H	R

解模糊器采用权重平均法将模糊输出变为明确的输出 LAD^*。

$$\mathrm{LAD}^* = \frac{\sum[\mu(\mathrm{LAD}) * \mathrm{LAD}]}{\sum\mu(\mathrm{LAD})} \tag{3-9}$$

在选定可以接入的小区之后，RAT 选择算法采用基于模糊 MADM 的 RAT 选择算法，从众多满足本地接纳条件的小区中选择最合适的 RAT。FB-JCAC 的主要特点是每

个本地 CAC 算法可以并行运行，执行速度快，扩展性好。

2. 基于负载均衡的联合接纳控制(LB-JCAC)

基于负载均衡的联合接纳控制是两层分级结构，分为水平呼叫接纳控制(HCAC)和垂直呼叫接纳控制(VCAC)。HCAC 以分布式结构分布在网络内，控制一个接纳网络内的接纳控制策略，如 WLAN 中的 AP、WCDMA 中的 Node B 等。VCAC 控制网络间的接纳策略，并进行集中部署。HCAC 和 VCAC 可以相互通信。根据分级描述，LB-JCAC 实体在异构网络环境中的部署如图 3-18 所示。

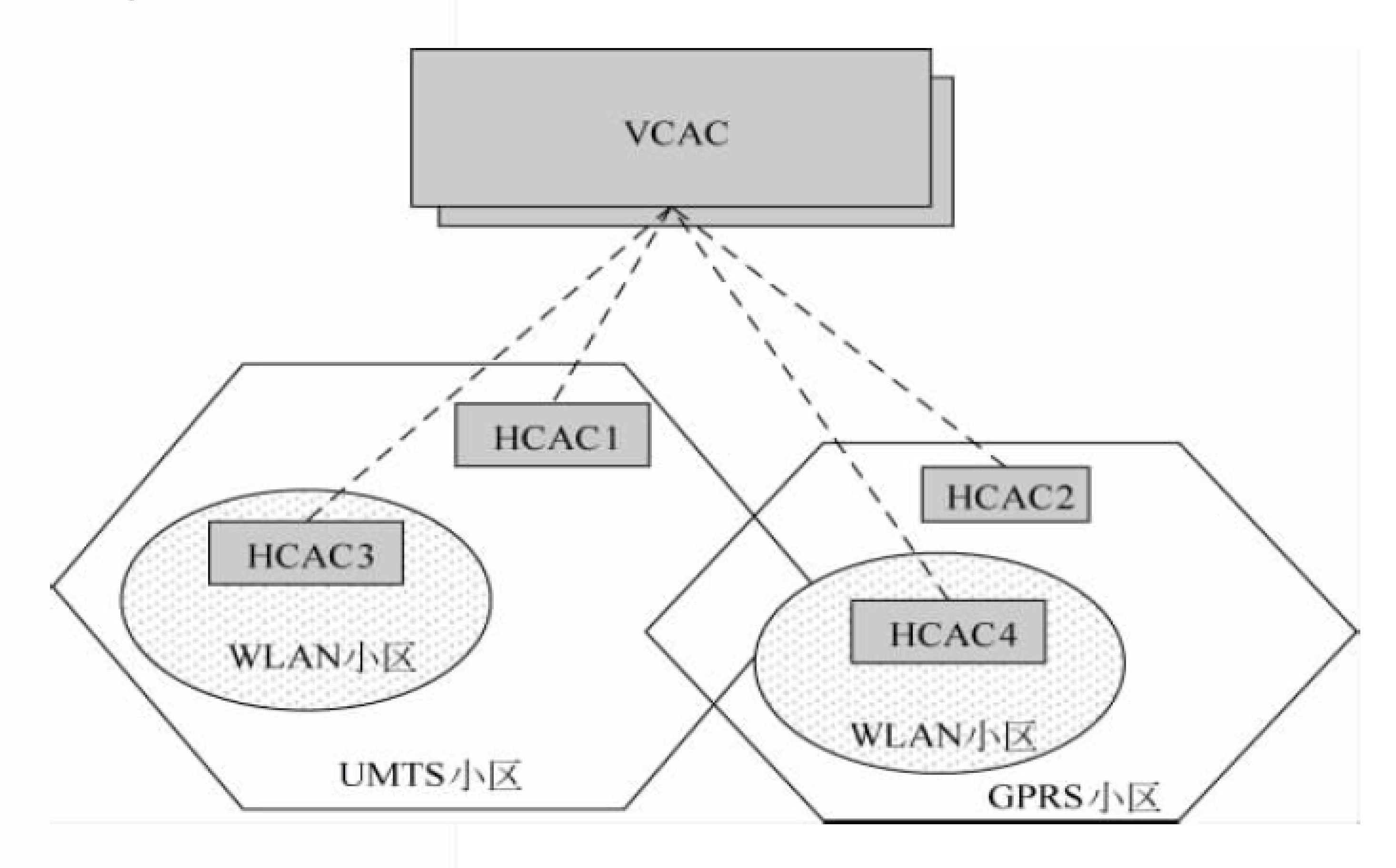


图 3-18 LB-JCAC 在异构网络环境中的部署

HCAC 的判决为三倍带宽预留机制(TTBR)。一种接纳网络的 C 通道被三个阈值 K_1、K_2 和 K_3 分为 4 个区域，其中 $K_1>K_2>K_3$，阈值是以服务提供者的参数选择为基础而设定的，如图 3-19 所示。当网络占有水平 L 低于阈值 K_3 时，语音和数据(不管是新的还是切换的)呼叫都能够被接纳进入网络。当 L 比 K_2 大时新的语音呼叫不会被接纳。如果 L 大于 K_1，垂直切换语音呼叫被丢弃。水平语音呼叫只有当 L 等于 C 的时候才被丢弃。

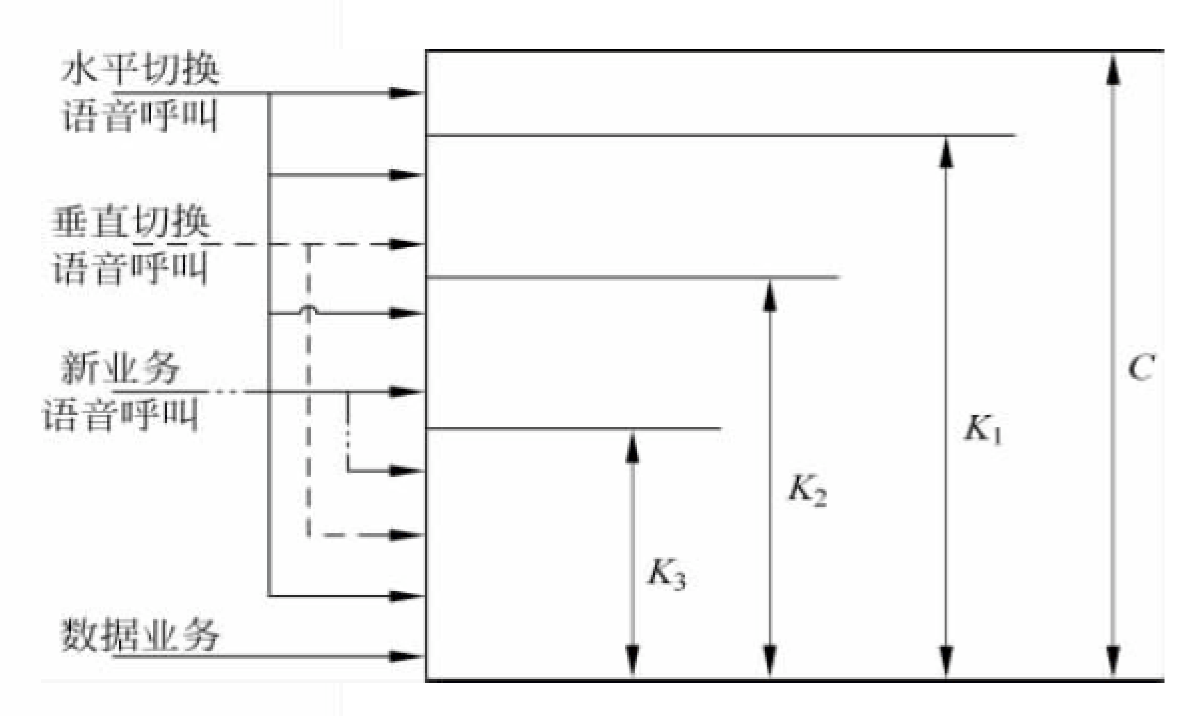


图 3-19 三倍带宽预留机制(TTBR)

当一个新的呼叫到达时，VCAC 从所有可选接入网络 HCAC 获取流量信息，从而判断哪个网络能够接纳呼叫和服务请求的参数选择是什么，通过比较得出最适合该呼叫的 RAT。

定义垂直负载级别 VLL。接入网的 VLL 就是某一类型的呼叫相比于其他接入网络所占用的带宽。使用 TTBR 判决时，VLL 的值 0、1、2、3 对应于 0、K_3、K_2、K_1。有两个 VLL 变量需要关注，当一个数据呼叫到达时，HCAC 只关注数据呼叫 VLL(d)。当语音呼叫到达时，它只关注语音呼叫 VLL(v)。

HCAC 周期性地向 VCAC 上报负载报告（LR），包括接入网络支持的业务类型、VLL 值和当前用户数。VCAC 接收到 LR 后，比较不同接入网络，根据当前用户数，VCAC 可以判定网络负载是否均衡，如果不均衡，就会做出相应调整。

3. 基于阈值的联合接纳控制（TJCAC）

本方案的使用场景是异构蜂窝网络。首先假定蜂窝网络如 GSM、GPRS、UMTS 等具有相同的覆盖范围且完全重叠，如图 3-20 所示。假定 RAT_j 内的带宽为 B_j，此处的带宽为等效带宽，可以为时隙、码序列等。这里提出了协同分布小区的概念，如图 3-20 中的小区 1a 和 2a 形成一个协同分布小区，1b 和 2b 形成另一个协同分布小区。

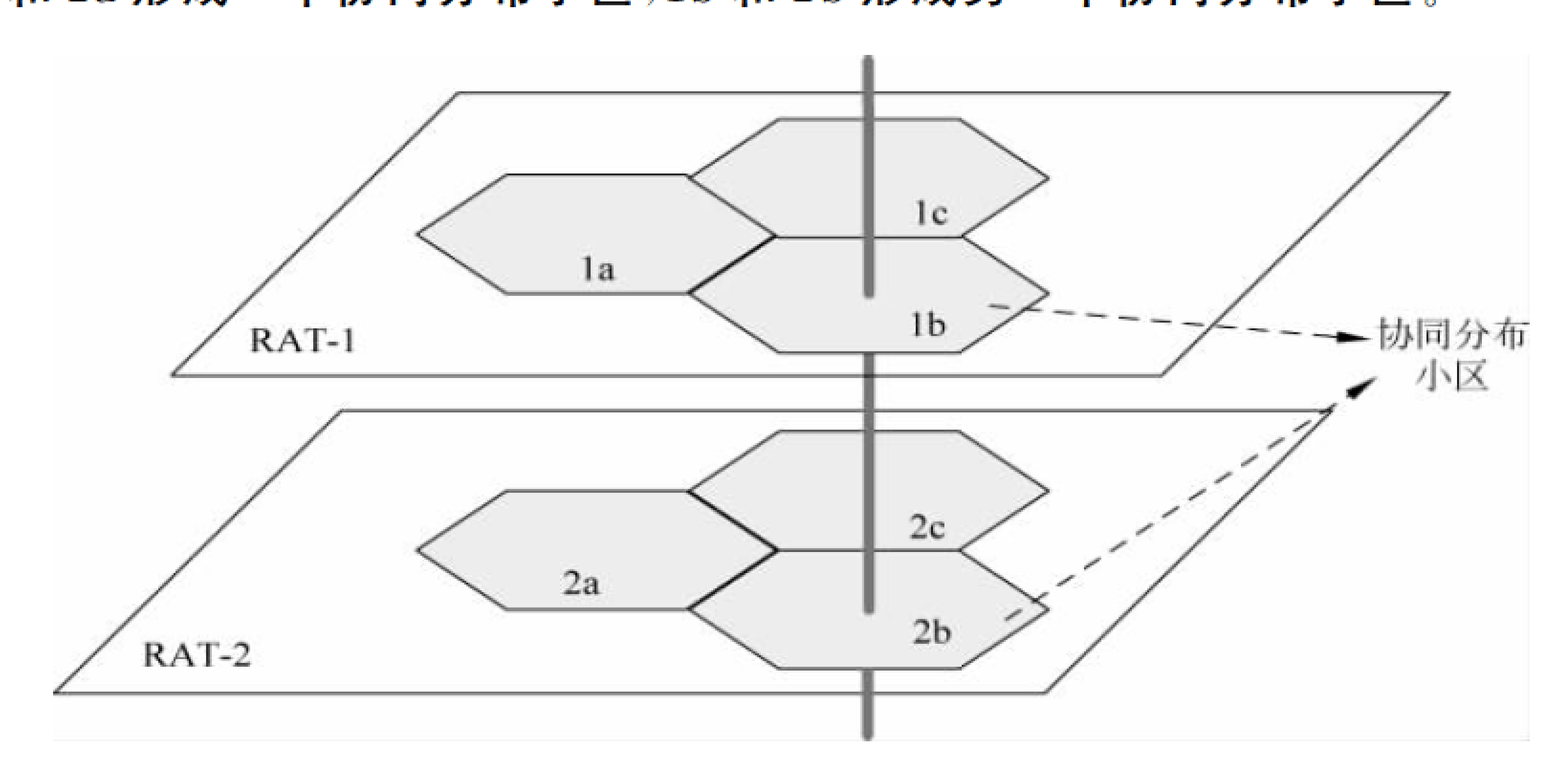


图 3-20 两种蜂窝技术场景下的协同分布小区

TJCAC 包括三个部分：联合接纳控制器、阈值带宽分配单元和带宽阈值更新控制器，如图 3-21 所示。在呼叫建立时，移动终端向联合接纳控制器发送业务请求。联合接

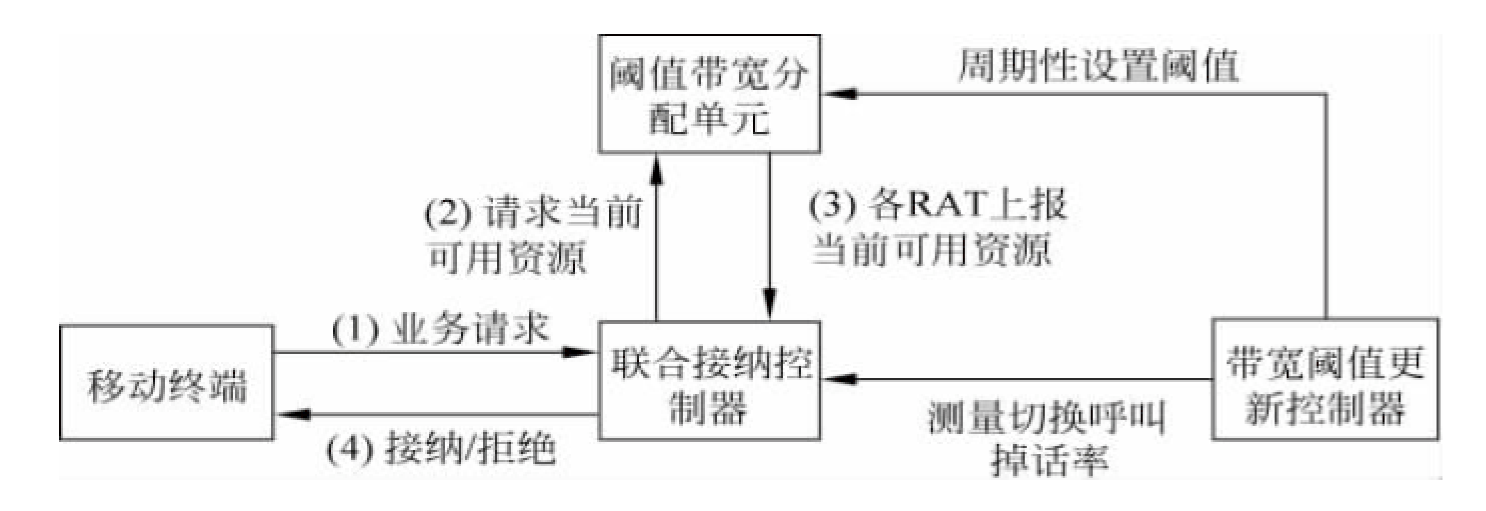


图 3-21 TJCAC 的组成

纳控制器从阈值带宽分配单元得到每个 RAT 的当前可用带宽，判决是否接纳该呼叫并且选择接纳的 RAT。

联合接纳控制器实施接纳控制算法。对于新呼叫来说，根据呼叫的带宽请求进行分类，$b_1 \leqslant b_2 \leqslant \cdots \leqslant b_k$。$C_{i,j}^n$ 代表 RAT_j 的等效带宽，$\alpha_{i,j}$ 为 RAT_j 的等效带宽占所有可用带宽的比重。对于切换呼叫来说，相应的参数为 $C_{i,j}^h$，$\beta_{i,j}$，则有

$$\alpha_{i,j} = \frac{C_{i,j}^n}{\sum^J C_{i,j}^n}, \quad \beta_{i,j} = \frac{C_{i,j}^h}{\sum^J C_{i,j}^h} \tag{3-10}$$

当新呼叫或切换呼叫到达协同分布小区时，JCAC 算法选择最轻负载的网络。假定 λ_i^n 和 λ_i^h 为新呼叫和切换呼叫的到达率，λ_{ij}^n 和 λ_{ij}^h 分别代表 RAT_j 中新呼叫和切换呼叫的到达率，则有

$$\lambda_{ij}^n = \alpha_{ij}\lambda_i^n, \quad \lambda_{ij}^h = \beta_{ij}\lambda_i^h, \quad \lambda_i^n = \sum_{j=1}^J \lambda_{ij}^n, \quad \lambda_i^h = \sum_{j=1}^J \lambda_{ij}^h \tag{3-11}$$

协同分布小区的业务到达过程分布如图 3-22 所示。

为了降低切换呼叫掉话率，阈值带宽分配单元为新呼叫和切换呼叫设置不同的阈值来区分优先级。带宽阈值更新控制器的思想来源于同构蜂窝网络。带宽阈值更新控制器周期性更新阈值 t_{ij}^n 和 t_{ij}^h，实时反应协同分布小区内的节点移动情况和网络流量现状。

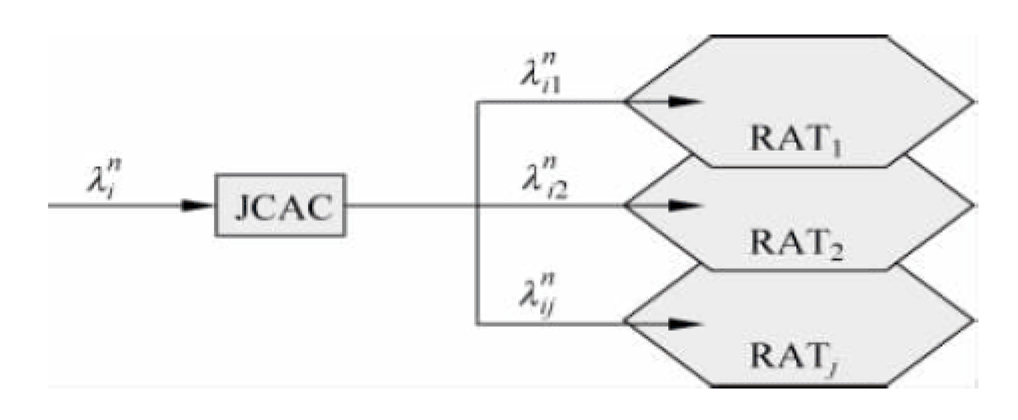


图 3-22　协同分布小区的业务到达过程分布

3.3　异构无线多接入网络选择

下一代无线通信系统将是一个能够将现有的和未来将要出现的各种无线接入系统有机融合在一起的开放式的网络环境，即异构无线网络。在这种场景下，对于一个用户来讲，它可能同时处在几种不同网络的覆盖下，为了保持通信的灵活性，未来的通信终端需要装备多个无线接口，这样通信的终端就可以根据当前环境灵活地选择接入的网络。随着可用网络的不断增加，接入选择变得非常复杂，需要高效的选择判决机制，以便对多种可用网络进行评估，从中选择最佳的接入，即 Always Best Connected[18]（简称 ABC 问题，即总是保持最佳连接）。

确定终端的接入目标的过程称为接入选择，这是异构无线网络中所特有的管理过程，也是异构无线网络能够满足用户个性化业务需求的关键所在。如果接入选择的决策不合理，不但会影响到用户个人的 QoS 水平，而且会影响到整个异构无线网络的资源利用率和运营商的商业利益。

接入选择作为实现 ABC 目标的主要功能之一，无论从用户的角度还是从运营者的角度来看，“总是保持最佳连接”在网络中都起着关键的作用。

接入选择算法为了充分挖掘并且综合利用异构无线网络中的集群增益和多接入分集增益，必须为每个多模终端中的每个应用进行合理的接入选择。作为保证异构无线网

络的无线资源得到充分利用的根本手段，接入选择算法在最近几年得到了越来越广泛的关注，并且已经成为异构无线网络研究中公认的核心问题和最热门的研究方向。以3GPP为代表的标准化组织以及各种科研机构都投入了大量的科研力量，并且已经取得了一定的研究成果。

3.3.1 无线多接入网络选择分类

1. 按照触发原因进行分类

根据触发原因的不同，接入选择分为初始接入选择和接入重选两种情况。

初始接入选择是对于新到达异构无线网络的无线应用而言，在建立初始的业务连接之前，必须调用某种接入选择算法来确定其初始接入系统，这种接入选择称为初始接入选择。而接入重选是对于那些已经完成了初始接入选择并接入到某个无线接入系统中的现有应用而言，当网络条件或者终端状态等因素发生变化时，则需要借助某种触发机制来确定是否有必要进行接入重选。

对于不同的异构无线网络，由于在网络框架、性能优化目标和接入选择模块的实现位置等方面存在差异，其初始接入选择和接入重选中所采用的接入选择算法既有可能是相同的，也有可能是不同的。

2. 按照异构无线网络中的实现位置分类

对于现有的各种接入选择算法，如果按其在异构无线网络中的实现位置来进行分类，可以分为集中式的接入选择算法和分布式的接入选择算法两大类。

集中式接入选择算法和分布式接入选择算法的优化目标不同：前者从无线运营商的角度出发，追求系统无线资源利用率的最大化，进而实现商业利益的最大化；而后者从用户的角度出发，希望以最小的代价、在最大程度上满足用户的个体需求。

1）集中式的接入选择

集中式的接入选择算法是在异构无线网络的网络方实现的，具体的实现位置与所采用的网络框架有关。在以紧耦合方式构成的异构无线网络中，接入选择模块位于主网络的无线资源管理模块中；而在以松耦合方式构成的异构无线网络中，接入选择模块位于核心网内的无线资源协同管理模块中。

对于初始接入选择而言，由于终端尚未接入到任何一个无线接入系统中，接入选择模块无法得到所需的用户信息，所以无法进行接入选择。因此，如果在异构无线网络中不存在从终端到接入选择模块的专用控制信道，就无法实现集中式的初始接入选择。对于这个问题，通常的解决方案是事先在终端中设定系统接入优先级，用户在初始接入时按照系统优先级从高到低的顺序依次进行接入尝试。实际上，这种优先级方案正是下一部分所要介绍的分布式接入选择算法中的一个简单实例。

在进行接入重选的过程中，多模终端需要向当前的接入系统报告相关的用户信息（例如用户的服务质量需求、接收信号强度等），再由该接入系统转发到位于主网络或者核心网中的接入选择模块；同时，各无线接入系统也需要将各自的系统信息（例如可用带

宽、系统载荷水平等）发送到接入选择模块中；最后，将接入选择模块所收集的用户信息和系统信息输入到事先定义的性能优化准则中，从而确定出多模终端的最佳接入选择，使得整个异构无线网络的无线资源得到合理利用。显然，上述过程对接入选择模块的数据处理功能和计算能力提出了巨大的挑战，也对接入选择算法的复杂度提出了严格的限定性要求。然而对于这些问题，目前还没有很好的解决方案，这也使得集中式接入选择算法的设计和实现成为异构无线网络研究中的一大技术难点。

2）分布式的接入选择

分布式的接入选择算法是通过设置在多模终端的接入选择模块实现的。在初始接入选择或者接入重选过程中，异构无线网络中的各个无线接入系统周期性地向多模终端发送广播信息（通常包含系统可用带宽、计费水平等内容），多模终端通过测量这些广播信息的信号强度来确定相应无线接入系统的可达性及其信道条件，同时从广播信息中获取各无线接入系统的管理信息。最后，在多模终端的接入选择模块中，将用户的业务需求和个人偏好、终端到各系统的信道条件和各接入系统提供的管理信息等输入到事先定义的接入选择准则中，从而确定出最佳的目标接入网络，使得终端能够在整个通信过程中总是保持最佳连接。

上述的接入选择过程无疑也是对终端的数据处理能力的一种挑战，不过随着多模终端功能的多样化和计算能力的日益强大，已经有可能将本来由网络节点承担的部分管理功能转移到终端来实现，从而一定程度上减小网络设备中的数据处理负担。

由于信令开销、信息安全以及一些与商业利益有关的非技术性因素等方面的原因，导致了网络方与用户方的信息不对称。而且，尽管在异构无线网络中多模终端具有了很大程度的自由选择权，但是无线运营商作为主体管理者的地位并没有改变。因此，在实际的网络运营中，上述两个优化目标通常无法同时实现。

实际应用中所采用的接入选择机制，往往是以某个优化目标为主体的运营商利益与用户利益的相互折中。例如，在进行集中式接入选择时增加关于用户个人需求方面的限定条件，从而在提高无线资源利用率的同时尽量照顾有特殊需求的用户；而在进行分布式接入选择时，在系统广播信息中引入有利于全局优化的指导信息（例如系统拥塞程度、拥塞价格等），从而引导用户向着有利于提高系统资源利用率的方向进行选择。

在一个具体的异构无线网络中应该选用何种接入选择算法，是无线运营商在综合考虑各种技术和非技术因素的基础上，利用细致的性能评估来决定的，无法笼统地说这两种算法孰优孰劣。

3.3.2 无线多接入系统的网络构架

无线多接入系统中网络选择技术的研究，与多接入技术的集成方式和多接入场景有着密切的关系。在不同的集成方式和场景下，多接入系统中网络选择的解决方案和算法有可能存在很大的差异。因此，作为前提和铺垫式的研究，需要全面考察未来无线多接入系统典型的集成方式，也就是如何在一个系统中集成多种无线接入技术，同时比较不同集成方式形成的多接入场景的差异。该部分研究使得多接入系统关键技术的研究具有更强的针对性。

对无线多接入系统的研究存在以下假设，这些假设是本节共同的前提条件：无线多接入系统整体特征符合 ITU 对 B3G 系统的特征描述；无线多接入系统中终端节点和网络节点的网络层协议采用 IETF 定义的 IPv6 协议；移动终端具备无线多接入能力，以及足够的运算速度和软硬件能力。该部分假设基于目前移动终端发展的趋势及未来简化网络智能、强化终端智能的需求。

1. 单一 RAT 接入网（S-RAN）

多接入技术（RAT）的集成可能发生在基础网络不同的接入层次上，据此将集成方式分为无线接入网间集成（inter-RAN integration）和无线接入网内集成（intra-RAN integration）。所谓无线接入网间的集成，是指无线接入技术单独组成无线接入网络，并通过各自的接入路由器接入 IP 核心网，网络结构如图 3-23（a）所示，称这种无线接入系统为单一 RAT 接入网（single-RAT RAN，简称 S-RAN）。由 S-RAN 组成的多接入系统，称为“S-RAN 多接入系统”。S-RAN 的无线接入部分彼此独立，其业务流分别通过各自的接入节点汇聚到骨干接入网络。目前，绝大多数商用的无线接入技术都是通过这种方式组成无线接入网络，如 WLAN 和 GPRS 系统。通常不同的 S-RAN 对应着不同的 IP 网段，也就是说，移动节点在跨越不同的 S-RAN 时，其子网前缀相应地会发生改变。

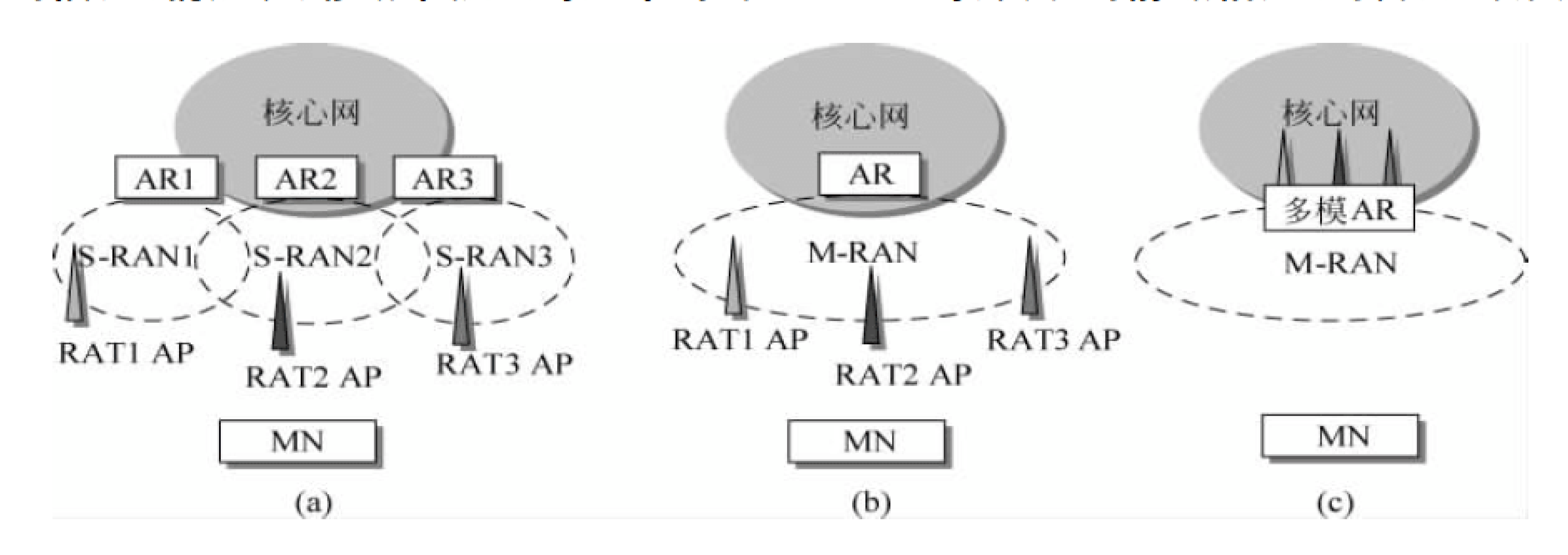


图 3-23　无线接入网络模型

2. 混合 RAT 接入网（M-RAN）

接入网内集成是指在一个无线接入网内集成多种无线接入技术，称这种无线接入网为混合 RAT 接入网（mixed-RAT RAN，简称 M-RAN）。M-RAN 的网络结构如图 3-23（b）和图 3-23（c）所示，图 3-23（b）中不同类型的二层无线接入点通过接入路由器和骨干接入网络连接。图 3-23（c）中 M-RAN 由网络侧的多模无线接入点（multi-mode access point）形成。一个 M-RAN 对应一个 IP 子网网段，移动节点在 M-RAN 移动的过程中，IP 子网前缀不会改变。

M-RAN 是一种全新的多接入形态，之所以提出这种更为紧密的网络层集成方式，有以下考虑。目前已经出现同时支持 802.11a 和 802.11b 的无线接入点，随着人们对无线多接入能力需求的增加，必然会形成对多模接入节点的应用需求。未来，M-RAN 可能成为一种广泛使用的便捷的无线接入网，其特点是在一个接入子网中包含了多种互补的无

线接入技术，如 WiMAX、WLAN、UWB 等，其业务流通过统一的接入节点汇聚到骨干接入网络。

这种网络典型的应用模型是具有多媒体业务需求的热点地区，如家庭、娱乐场所、办公区的大型多媒体会议室等。这种新型的混合无线接入网络在热点地区提供了多个同时可用的互补的无线接入链路，同时使用多无线接入能力会给用户提供更为多样和高速的业务感受，从而形成便捷的局域无线多媒体网络。

3. 无线多接入场景

上述两种无线接入网重叠覆盖会形成三种典型的无线多接入场景，如图 3-24 所示。

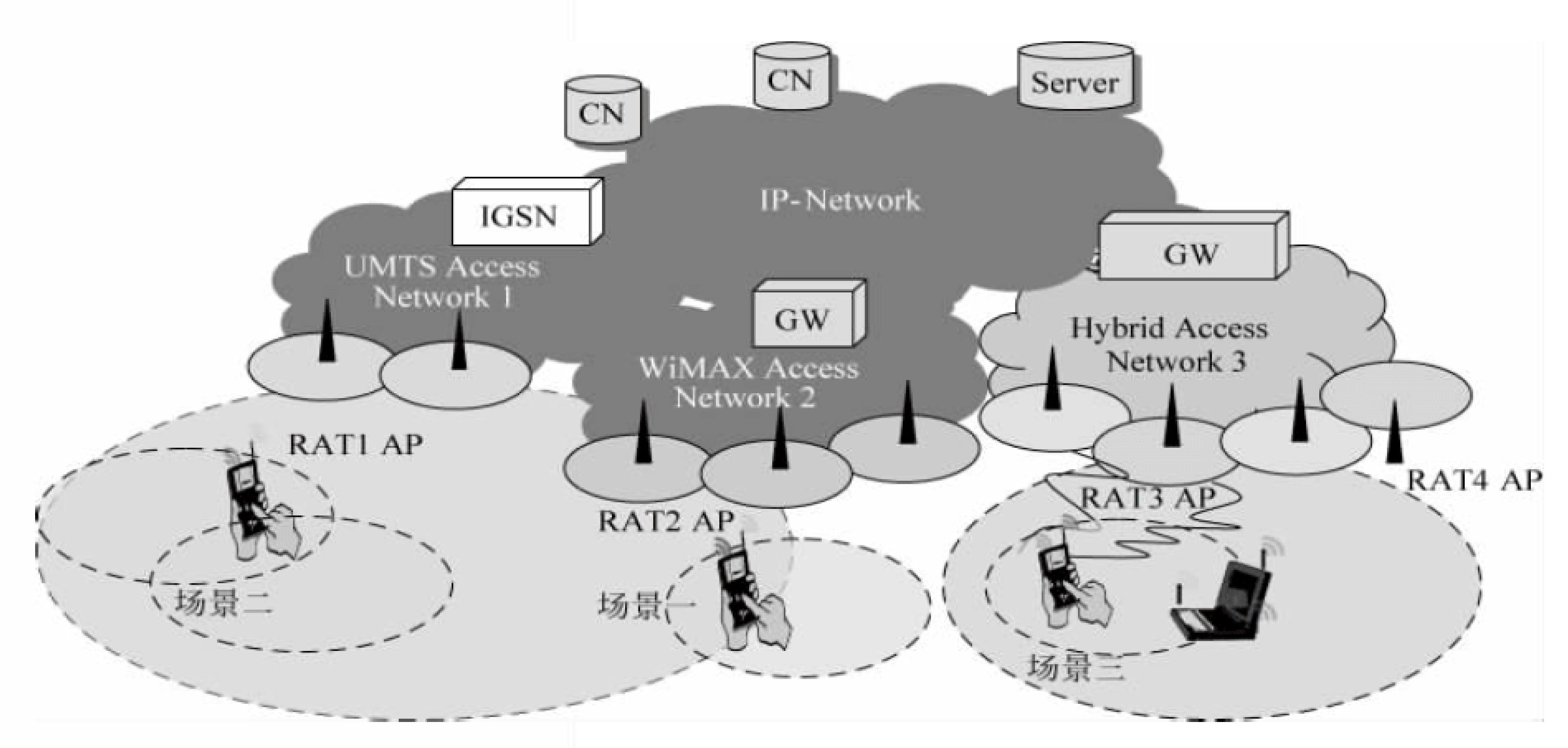


图 3-24 典型的多接入网络场景

场景一，S-RAN 边缘重叠覆盖。移动节点处于多个异种 S-RAN 重叠覆盖的区域，其重叠区域由多个小区的边缘重叠覆盖形成；其特点是重叠覆盖区域中存在多个可用的无线链路，但是由于是网络边缘的重叠覆盖，数据传输质量都不太好。这种场景的典型代表是热点区域和非热点区域的结合部，如某运营者在热点区域铺设大容量电信级的 3G 网络，而在非热点区域使用固定无线接入技术如 WiMAX，并在 3G 和 WiMAX 网络形成的多接入系统范围内，提供业务漫游和业务保持的能力。这种网络场景便于节省设备投资、扩大网络的覆盖范围，具有现实意义。

场景二，S-RAN 嵌套重叠覆盖。移动节点处于两种 S-RAN 完全重叠覆盖区域，和场景一不同的是多个异种小区呈现嵌套式重叠覆盖，移动节点具有两个以上稳定的无线链路。这种场景被形象地称为 Pizza 模型，其典型代表是热点区域。

场景三，M-RAN 嵌套重叠覆盖。移动节点处于同一个 M-RAN 完全重叠覆盖区域，具有两个以上稳定的无线接入链路，且不同的无线接入链路归属于同一个无线接入网络。这种场景的典型代表是前面提到的家庭或娱乐热点区域。如果单纯从小区形成的重叠覆盖情形看，场景二和场景三没有太大区别，但是，由于网络侧集成模型的不同，这两种场景下的移动性管理和多链路并行传输的解决方案也完全不同。

关于无线多接入系统关键技术的讨论将针对上面三个典型的多接入场景分别展开。

这三个场景间所存在的差异,决定了解决方案的差异性。表 3-3 详细列出了三个场景之间的差别。

表 3-3　典型多接入网络场景间的比较

场　　景	场　景　一	场　景　二	场　景　三
RAN 类型	异种 S-RAN	异种 S-RAN	M-RAN
重叠覆盖	异种小区边缘重叠	异种小区嵌套重叠	异种小区嵌套重叠覆盖
集成层次	网络层	网络层	网络层或链路层
网络层地址数	多个 IP 地址	多个 IP 地址	一个或多个 IP 地址
移动性支持	Mobile IP	扩展的 Mobile IP	None
多路径传输时延	差别较大	差别较大	小,仅无线链路一跳

3.3.3　多接入选择过程

网络选择是一个复杂的问题,通常情况下网络选择分为 4 个步骤,如图 3-25 所示。

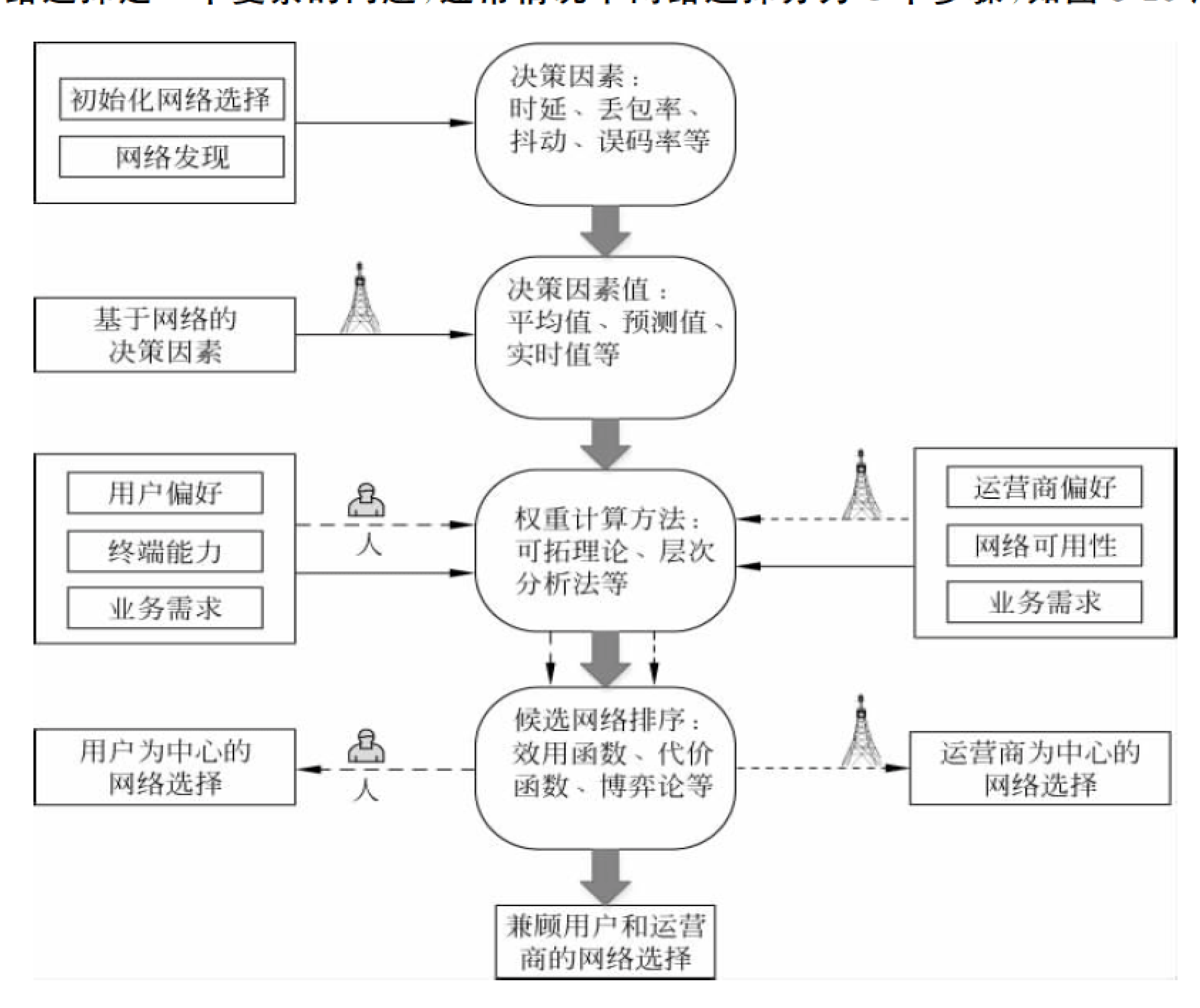


图 3-25　网络选择过程(引自参考文献[11])

在网络选择过程中并不是必须包括这 4 个步骤,这取决于所应用的选择机制。但在大多数情况下,每个步骤都代表网络选择期间不同的过程[11]。具体内容如下:

(1) 决策因素的选择。这一步指的是确定在决策过程中需要考虑的所有参数。

（2）决策因素值的收集。在确定所有决策因素后，通过候选网络收集每个决策因素的值，为了便于比较分析，收集后进行规范化处理。

（3）决策因素权重的估计。确定所选决策因素的权重，每个决策因素的权重值体现了该因素对于产生网络选择最终结果的重要性，所有决策因素权重值的总和必须为1。

（4）候选网络的排序。最终，根据每个候选网络的性能值的大小完成候选网络的一个排序，从中选取最优网络。

3.3.4 基于效用函数的接入选择算法

在微观经济学的意义上，效用是用户对所感知到的服务质量的满意度的测量。不同形式的效用函数表现不同弹性流量的特色。在这种情况下，用户 i 的效用函数常用 $U_i(R_i)$ 描述，该函数取决于“有效的传输速率”这一变量。对于简单的分析，可以忽略掉一些物理层的因素，如信道衰落和信号衰减[19]。

$U_i(.)$ 的形式应该满足以下两个特点以代表实际的满意程度，现提出以下假设[20]：

（1）$U_i(.)$ 是一个单增、严格凹和可两次微分的函数，且满足 $U_i(0)=0$，上限为 $U_{i\max}$，即用户 i 可达到的最大利用率。

（2）存在常数 K_i，对于所有的 R_i 都有 $U'(R_i)$ 小于等于 K_i。

在一些文献中，针对不同流量的效用函数已经进行了一定的研究。例如，在参考文献[21]中，对语音、视频和数据三种不同类型的应用，分别提出效用函数。假定语音服务为CBR（恒定比特率）的业务，其在传输速率方面的QoS要求是严格的，所以可以由一个阶跃函数代表。为了满足假设①，在语音服务中可以用一个尖锐（高斜率）的凹函数代替阶跃函数。数据服务被认为是一种ABR（可用比特率而定）的贪婪业务，这意味着更多的传输速率会得到较高的满意程度，可以用一个递增的凹函数形容其效用。视频服务被假定为VBR（可变比特率）的业务，其QoS的要求在语音和数据间，所以由一个斜率在语音和数据之间的凹效用函数代替。

1. 市场模型

假设一区域被几个网络覆盖，如WLAN、WCDMA和GPRS等。当用户进入该领域后，它的多模终端可以接收到由WLAN的接入点、WCDMA和GPRS的基站发送的几种信号，每个用户或上层应用根据网络资源的可用性、业务特性、无线链路质量、用户偏好等预定一种无线网络。

令 $y=\{1,2,\cdots,Y\}$ 为一组无线网络，$s=\{1,2,\cdots,S\}$ 为一组由这些网络支持的服务。考虑这些无线网络中的一种，则向量 $N_s=\{N_1,N_2,\cdots,N_s\}$ 代表当前时刻每种服务的活动用户数，向量 $R_s=\{R_1,R_2,\cdots,R_s\}$ 代表每种服务所对应的平均数据速率。R 是网络的最大约束容量。$U_s(R_s)$ 表示业务 S 的效用函数。为最大化所有用户的总效益可以推导一个最佳问题，如下：

$$
\begin{aligned}
&\max_{R_s} \sum_{s=1}^{S} N_s U_s(R_s) \\
&\text{s.t.} \sum_{s=1}^{S} N_s R_s \leqslant R
\end{aligned}
\qquad (3\text{-}12)
$$

因为所有的效用函数都被假定为严格凹函数，所以对于速率向量 R_s 必有一个唯一的最优解，该解可由拉格朗日方法得出。拉格朗日函数被定义为

$$L(R_s,\lambda)=\sum_{s=1}^{S}N_sU_s(R_s)-\lambda\left(\sum_{s=1}^{S}N_sR_s-R\right)$$
$$=\sum_{s=1}^{S}N_s(U_s(R_s)-\lambda R_s)+\lambda R \tag{3-13}$$

其中，λ 是拉格朗日的乘数，假设 $R_s^*=\{R_1^*,R_2^*,\cdots,R_s^*\}$ 是最适宜的速率向量，利用参考文献[6][7]中的相似方法。

$$R_s^*(\lambda)=\underset{R_s}{\operatorname{argmax}}\,L(R_s,\lambda) \tag{3-14}$$

为了解决以上问题，先定义用户收入函数

$$B_s(R_s,\lambda)=N_s(U_s(R_s)-\lambda R_s) \tag{3-15}$$

则式(3-14)可以用如下形式表示：

$$R_s^*(\lambda)=\underset{R_s}{\operatorname{argmax}}\,B_s(R_s,\lambda)\quad(s=1,2,\cdots,S) \tag{3-16}$$

因此，网络的总效用最大化问题被转化为最大化用户收入子问题。

另外，如我们所知，当 $\sum_{s=1}^{S}N_sR_s=R$ 时，式(3-12)中最大化问题的最优解决方案存在，其中 R 是网络能提供的最大无线资源，并且 $\sum_{s=1}^{S}N_sR_s$ 是用户消耗资源的要求。

2. 用户收入最大化和网络间的市场均衡

根据微观经济理论，如果在市场中需求以价格 X 满足供应，这是所谓的市场均衡，相应的价格 X 是均衡价格。

正如上面所讨论，通过在用户和网络间引入价格，市场模型可分为两个子问题。第一个子问题是如何最大化每个用户的收入。一个有业务 s 的用户按价格 X 付费并可以自由调整传输速率 R_s。用户收入最大化问题为

$$\begin{aligned}&\text{User}(B_s(R_s,\lambda)):\\&\max_{R_s}B_s(R_s,\lambda)\quad(s=1,2,\cdots,S)\end{aligned} \tag{3-17}$$

需要注意的是，在此问题中用户面对得到高效用(通过选择一个高传送率)和支付低消费(通过选择一个低传送率)两者间的折中。如果效用函数的假设(1)成立，解决方案必须满足第一条件

$$\frac{\partial B_s(R_s,\lambda)}{\partial R_s}=0 \tag{3-18}$$

或

$$U_s'(R_s)-\lambda=0$$

接下来，定义需求函数 D_s 为

$$D_s(\lambda)=R_s(\lambda)\quad(s=1,2,\cdots,S) \tag{3-19}$$

$D_s(\lambda)$函数则为求式(3-17)最大化问题的最优解 $R_s^*(\lambda)$。$D_s(\lambda)$可以理解为当价格为 λ 时业务 s 要求的传送速率。

进一步，如果假设(2)成立，对于业务 s 存在一个常数 λ_{smax}，使得

$$D_s(\lambda) = 0 \quad \forall \lambda \geqslant \lambda_{smax} \tag{3-20}$$

λ_{smax} 称为业务 s 的保留价格，代表着业务 s 可以支付的最高资源价格。或者说，当价格超过 λ_{smax} 时，业务 s 对传输速率的要求会变为零。

第二个子问题是在网络端解决市场均衡的问题。目标是令资源价格 λ 收敛到均衡价格 λ^*，以达到供求平衡并最大化网络的利用。

$$\lim_{\lambda \to \lambda^*} \sum_{s=1}^{S} N_s R_s = R \tag{3-21}$$

为了获得均衡价格 λ^*，可以采用半折查找法。在市场模型中，价格 λ 代表着网络条件，价格随负载变重而升高(抑制用户消费)，随负载减轻而降低(鼓励用户消费)。

如前面讨论中提到的，最佳资源分配可以通过每个用户最大化自己的收入得到，这是一个分布的、并行的计算过程。因此，中心控制节点如 AP 或 BS 的计算复杂度减轻，可以使该机制适合应用到真实的系统中。

3. 算法描述

本部分主要介绍三个算法：均衡价格算法、网络选择算法和请求接纳控制算法。每个终端的信号检测模块被认为可以同时检测到各个网络的信号。每个网络的无线资源管理模块存储一些系统、业务参数，如网络容量限制 R、效用函数 $U_s(R_s)$、需求函数 $D_s(\lambda)$ 和每个业务的保留价格 λ_{smax}。

另外，不同网络有适合不同业务的特性。例如，用户喜欢通过接入 WLAN 下载大数据文件，但会选择蜂窝网络用于执行语音通话。定义 $p_{sx}(0<p_{sx}<1)$ 用于描述用户偏好，其中 s 和 x 分别代表不同的服务类型和网络。

1) 均衡价格算法

既然价格 λ 可以表示网络资源的利用率，设计这个均衡价格算法的目的是动态地调整价格 λ 以趋近均衡价格 λ^*，均衡价格 λ^* 可以使网络利用率达到最大。该算法的收敛精度被定义为 ε。

(1) 初始化资源价格。网络初始化时定义 λ_{max} 和 λ_{min} 为最高和最低的可能价格，并设置初始价格 $\lambda_{initial}=0.5(\lambda_{max}+\lambda_{min})$，然后广播这个初始化的资源价格。

(2) 调整传输速率。每个用户根据广播的资源价格 $\lambda_{initial}$ 和需求函数 $D_s(\lambda)$，调整传输速率 R_s。

(3) 调整价格。网络做一个全网吞吐量 $\sum_{s=1}^{S} N_s R_s$ 的调查，当 $\sum_{s=1}^{S} N_s R_s != R$ 或 $\lambda_{max}-\lambda_{min}>\varepsilon$ 时，调整价格。如果 $\sum_{s=1}^{S} N_s R_s < R$，则使 $\lambda_{max}=\lambda_{current}$ 降低价格，或使 $\lambda_{min}=\lambda_{current}$ 提高价格。然后，更新当前价格 $\lambda_{current}=0.5(\lambda_{max}+\lambda_{min})$；广播当前资源价格 $\lambda_{current}$；返回第二步，最后广播均衡价格 $\lambda^*=\lambda_{current}$。

2) 网络选择算法

网络选择算法是旨在引导用户行为，根据用户使用偏好参数减少网络拥塞和切换过程中的乒乓效应。用户终端根据 RSS(接收信号强度)、资源价格、用户偏好选择最有效

的网络。三个预先定义的参数集存储在用户终端中：$y=\{1,2,\cdots,Y\}$是无线网络集，$x=\{1,2,\cdots,X\}$是候选无线网络集合，RSS 门限集合是 $T=\{T_1,T_2,\cdots,T_Y\}$（RSS 高于此网络的门限）。

（1）检测 RSS。

```
For y = 1: Y，在一个长度为 T1 的平滑窗口中检测网络 Y 的 RSS，并计算 RSS(y)的平均值；
If(RSS(y)> T(y))，网络 y 是候选网络，把它加到 x 集合中来；结束。
```

（2）选择网络。选择候选网络集合中最便宜的一个网络，用 $x_{\min}=\text{argmin}\lambda_x$ 表示，把它的价格记为 $\lambda_{x\min}$。

```
If 请求是新的；
接入网络 x_min，x_access = x_min；
Else 请求是一个传递请求；
If(x_current = x_min)；
Break；
Else 计算两个网络在价格上的相差比 v = (λ_current − λ_xmin)/λ_current；
根据服务 s 选择用户偏好参数 P 和目标网络 x_min，决定以 p_sx 的可能性切换到目标网络 x_min；
End
End
```

3）请求接纳控制算法

呼叫接纳，是基于保留价格 $\lambda_{reservation}$ 的概念，这可以保证已建立连接终端用户的 QoS。有三种类型的呼叫：新的呼叫、价格切换呼叫（切换是由资源价格引起）和移动切换呼叫（切换是由移动性引起的，例如用户从 WLAN 漫游到无线广域网）。接入时不同的呼叫有不同的优先权。请求接纳控制算法步骤如下：

（1）估计影响。计算网络接纳新呼叫或切换呼叫后的临时资源价格 λ_{temp}。

（2）比较价格，估计影响。

```
If λ_temp< λ_reservation
接纳所有呼叫；
Else
拒绝新呼叫；
重算如果网络接纳所有切换呼叫后的临时资源价格 λ'_temp；

if λ'_temp< λ_reservation
接纳所有切换呼叫；
Else
拒绝价格切换呼叫；
重算网络接纳所有移动切换呼叫后的临时资源价格 λ''_temp；

if λ''_temp< λ_reservation
接纳所有移动切换呼叫；
Else
拒绝移动呼叫；
End
End
End
```

3.4 异构无线网络端到端 QoS 保障

本节首先关注提高异构网络 QoS 的三种服务模型，即基于资源预留的集成服务（IntServ）、基于优先权机制的区分服务模型（DiffServ）和多协议标签交换模型（MPLS），然后讨论异构网络的 QoS 映射。

3.4.1 IntServ 集成服务模型

Internet 最初是面向非实时的、单一数据类型通信的，由于设计时没有关注 QoS 控制和管理技术，因此资源完全共享，资源的访问和使用没有有效的控制和管理手段，不能保证用户的 QoS 需求。IETF 借鉴 QoS 技术，加强实现资源的控制和调整机制，使得网络能够支持多种服务，提出了一种集成服务体系 IntServ。

IntServ 的目的是在基于 IP 的网络中提供服务质量支持。IntServ 力图解决在网络发生拥塞时如何共享可用的网络容量的难题。集成服务以 RSVP 信令向网络提出业务流参数，并建立和拆除传输路径上的业务流状态。主机和路由器节点建立和保持业务流状态信息。尽管 RSVP 经常用于单个数据流，但也可以用于聚集流的资源预留。

1. 体系结构

在 RFC 1633 中，指出了 IntServ 整体解决方案所包含的 QoS 组件，网络中的每个路由器都需要实现[21]，如图 3-26 所示。

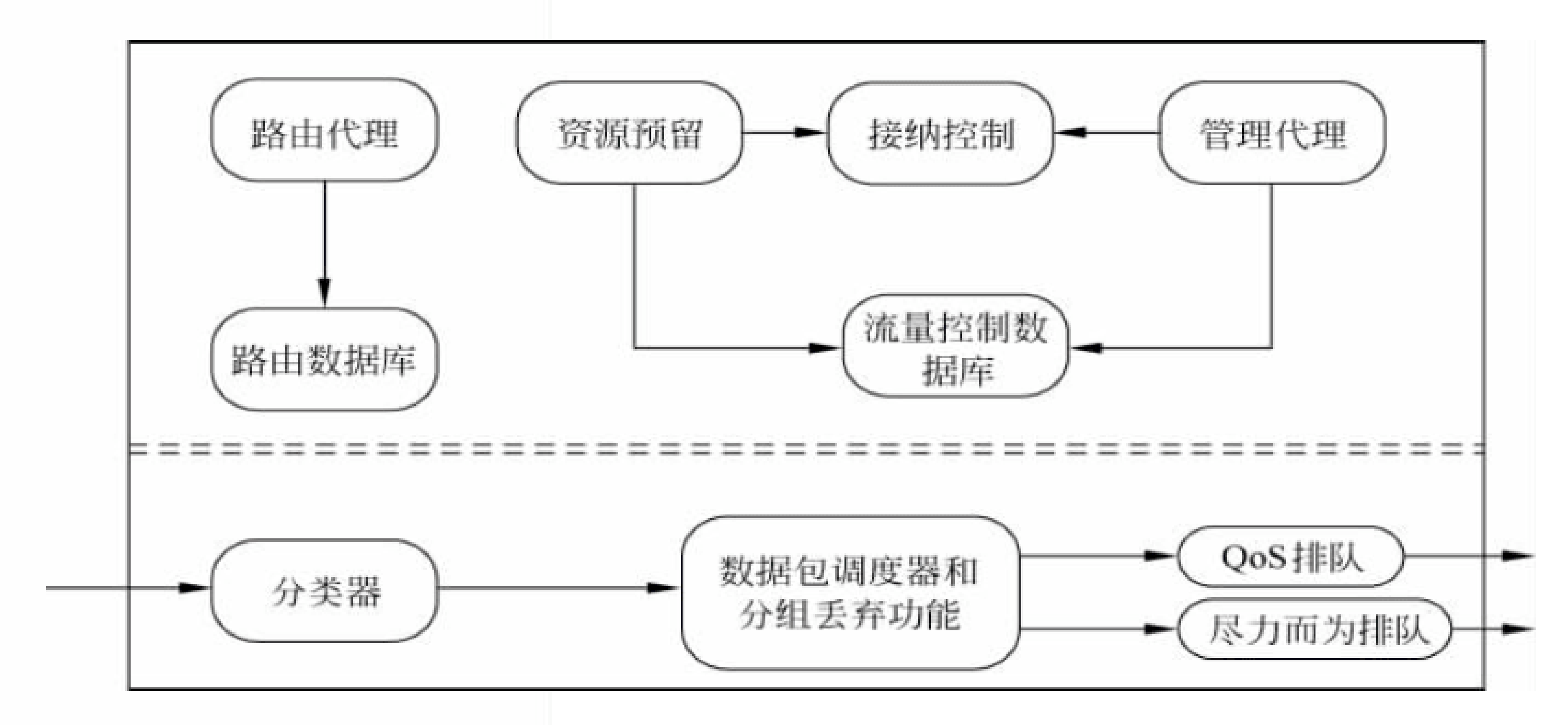


图 3-26　IntServ 模型的 QoS 组件

（1）资源预留协议（RSVP）。RSVP 是 Internet 上的信令协议[23]，也是 IntServ 的核心技术。通过 RSVP，用户可以为每个业务流或连接申请预留资源，路径上的每一个节点都要进行预留操作，从而提供端到端的 QoS 保证。RSVP 是单向的预留，适用于点到点及点到多点的通信环境。

（2）接纳控制。当一个新业务流的资源预留请求到达时，预留协议就调用该模块来判断是否有足够的资源满足该流所请求的 QoS，其他这个判断是根据当前已对业务流的

预留承诺和网络的当前状况而做出的。

(3) 管理代理。监督接纳控制模块并设置接纳控制的策略，同时它能够修改业务控制数据库，以影响队列调度和分组丢弃。

(4) 路由代理和路由数据库。对 IP 分组进行分类，根据最小代价及其他服务质量参数进行路由选择，维护一个路由数据库，对每个目的地址和流都给出下一跳地址。

(5) 分类器。根据预置的规则对进入路由器的每个分组进行分类。分类后的分组被放到不同的队列中等待接收服务。

(6) 队伍调度器和分组丢弃功能。队伍调度器主要是基于一定的调度算法对分类后的分组队伍进行调度服务。常见的调度算法有 WFQ、WF2Q、SCFQ、VC、MD-SCFQ 和 WRR 等。分组丢弃功能主要基于 WRED、RIO 策略算法，当网络发生阻塞时进行数据包丢弃。

在集成服务体系中，欲发送分组的节点首先向网络发出发送申请，申请是通过信令来完成的，即 RSVP 协议。应用程序通过信令协议通知网络待发送分组所需的 QoS 参数，包括带宽、时延、丢包率等。收到资源预留请求后，传送路径上的网络节点实施接纳控制，验证用户的合法性并检查网络资源的可用性，最终判定是否接纳新业务并为其预留资源。一旦接纳申请得到批准，并为流量分配了网络资源，则只要提交的流量在网络运营者和用户事先约定的参数范围之内，网络就应按照承诺的 QoS 需求提供服务，预留路径上的网络节点可以通过执行分组的分类、流量管制、低时延的排队调度等 QoS 控制机制，来实现对承载流量的 QoS 保证。

2. 业务分类

在 IntServ/RSVP 业务流中，定义了三种类型的业务，如表 3-4 所示。

表 3-4 三类业务与编码、优先级别的对应关系

编　码	优先级别	说　明
111	1	保证型服务
110	2	受控负载服务
101	3	
100	4	
011	5	
010	6	
001	7	
000	8	尽力而为服务

(1) 保证型服务(guaranteed service，GS)。RFC2212，具有明确的参数级别，要求先设定网络传输的最大延时和相应的带宽。然后根据设定参数，在源节点到目的节点之间建立连接，连接后，保证在传输过程中业务流所需的时延和带宽。它的码值是 111，其优先级别为 1(最高)。实际上它是一种高速的专线服务业务。

(2) 受控负载服务[24](controlled load service)。RFC2211，该业务流没有固定的时延保证，不设定最大的传输延时，但需要一定的、长期的带宽保证，它只保证网络的负载无论怎样变化，都能对受保护的业务流通过改变所需参数(可降低)提供相应的资源预留

保障。它的码值是 110～001，其优先级别为 2～7。某节点发生拥塞时，该服务是通过丢弃部分优先级别低的信息包来保证特殊的业务不受影响的。

(3) 尽力而为服务(BE)。传统 IP 网络的传输方式。它没有带宽和延时的保证，业务只是基于可达性。当网络较为宽松时，用户能获得较好的服务；当网络拥挤时，用户的 QoS 也随之下降。它的码值是 000，其优先级别为 8(最低)。

三类业务与编码、优先级别的对应关系如表 3-4 所示。

IntServ 属于细粒度 QoS 控制，即可以为高层用户提供较为准确的带宽、延迟、延迟抖动和丢失率控制，能够提供有端到端 QoS 保证的传送服务。通过 RSVP 能够为不同的优先级别申请不同的资源预留，因为 RSVP 运行在从源端到目的端的每个路由器上，因此可以监视每个流，以防止其他未申请业务流的资源占用；另外，核心技术 RSVP 使用 IP 包承载，使用“软状态”的概念，随时可以更改该预留的通道和资源预留的释放，灵活性较强。

但在具体的实现中 IntServ 还存在扩展性较差的缺点，因为 IntServ/RSVP 要求端到端的信令，这就必须要求从发送者到接收者之间的所有路由器都支持所实施的信令协议，并且每个路由设备都要求对 RSVP 处理，同时保存流状态信息。其结果是消耗大量网络资源，造成整个大型网络负担过重。在网络规模扩大时，相应的维护开销会大幅度增加，对路由器特别是核心路由器线速处理分组的性能造成不良影响，不适于在流量汇集的骨干网上大量应用。并且每个 QoS 协商过程都要有比较大的延迟，当流的数量增大后，容易造成网络拥塞。另外，IntServ 着眼于保障多媒体应用而不是非多媒体应用，但实际上后者仍是网络业务中很重要的一部分。总之，这些原因都限制了 IntServ/RSVP 更广泛的应用。

3.4.2 DiffServ 区分服务模型

在 IntServ 体系的发展遭遇巨大障碍的时候，DiffServ 服务模型应运而生，最早是在 IETF RFC2475 中进行定义的。DiffServ 将复杂的流分类和流量控制交给边界路由器完成。边界路由器负责完成复杂的流分类、为分组打 DSCP 标记、流量的接入速率管制和访问控制等动作。区域内部的路由器只需进行简单的流分类，对 BA 实施流量控制。这样就避免要求 IntServ 模型的核心节点支持基于某个流的复杂的流分类及实施流量控制，从而使 DS 网络内部的转发操作可以更高效地实现。DiffServ 具有简单有效、扩展性强的优点，克服了集成服务模型(IntServ)的可扩展性问题，是可以满足多种业务服务需求的 IP QoS 模型。

1. DiffServ 的体系结构

如图 3-27 所示，一个完整的 DiffServ 体系结构包括[21][25]：

(1) DiffServ 节点。实现 DiffServ 功能的网络节点称为 DS 节点，分为 DS 边界节点和 DS 内部节点，DS 边界节点将 DS 域和其他 DS 域或非 DS 域连接在一起，根据域间指定的流量调节协议 TCA(traffic conditioning agreement)进行流量控制；DS 内部节点仅负责在同一个 DS 域内连接 DS 边界节点和其他内部节点，基于 DSCP 进行简单的流分类以及对 BA 实施流量控制。两种节点都可以根据分组的 DSCP 选择相应的 PHB 进行转

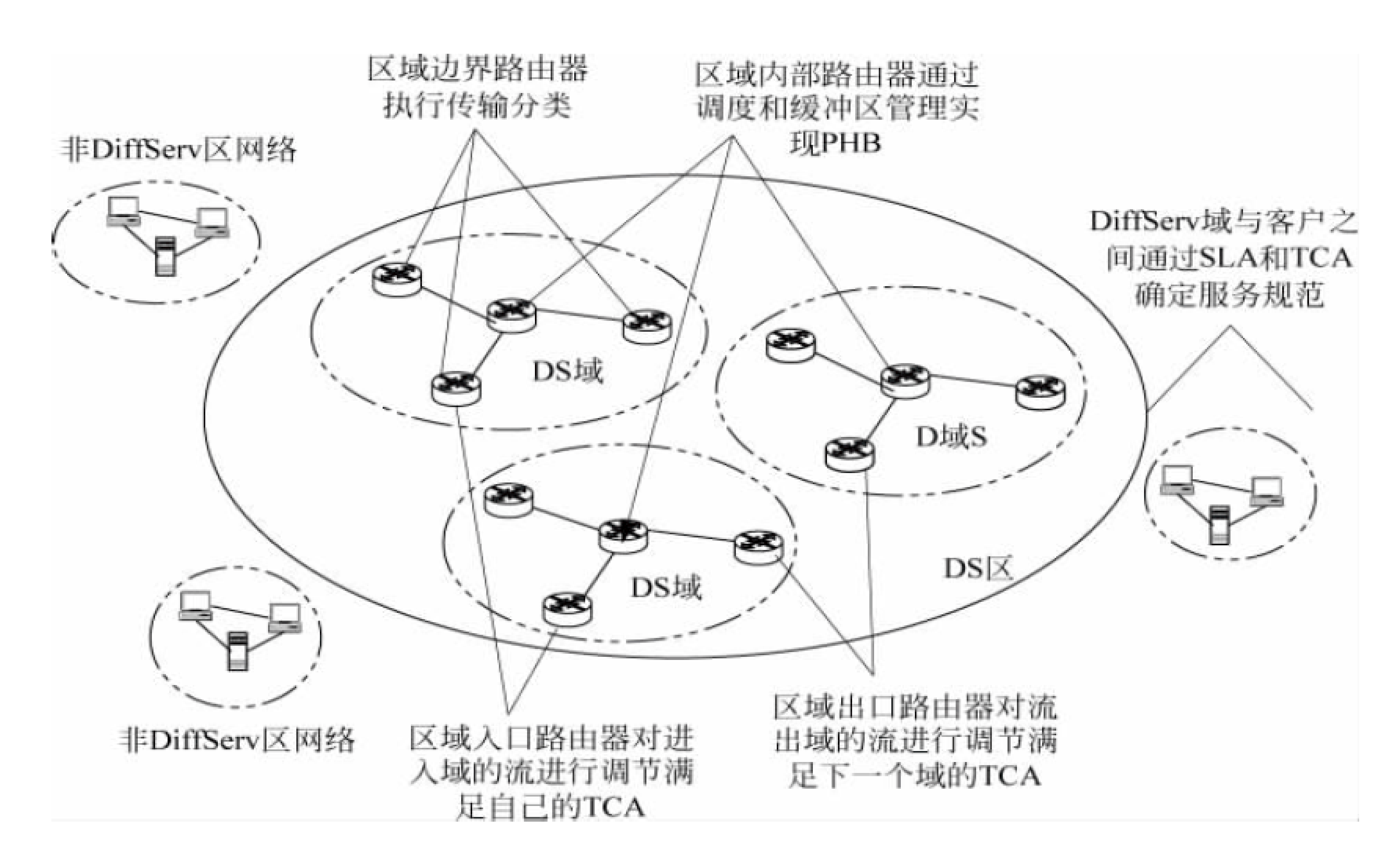


图 3-27 DiffServ 的体系结构

发操作。

(2) DS 域(DiffServ Domain)。由一组采用相同服务提供策略和实现相同 PHB 组集合的相连的 DS 节点组成,DS 边界节点同时作为 DS 域的输入和输出节点,分组从输入节点进入 DS 域,从输出节点流出 DS 域。

(3) DS 区(DiffServ region)。一组相互毗邻的 DS 域可组成 DS 地区。

与 IntServ 不同,DiffServ 采用聚合的机制将具有相同特性的若干通信流聚合起来,为整个聚合流提供服务,而不再面向单个通信流。也就是说,在 DiffServ 网络边界路由器上保持每个业务流的状态,核心路由器只负责数据包的转发而不保持状态信息。其基本实现方法是:

(1) 简化网络内部节点的服务机制。在内部节点只进行简单的调度转发,网络内部的核心路由器中只保存简单的 DSCP(diffServ code point)与 PHB(per-hop behavior)的对应机制,在数据流进入核心路由器时只根据数据包头部 DS(differentiated services)域中的 DSCP 进行转发,而业务流状态信息的保存与流监控机制的实现等只在边界节点进行,内部节点是状态无关的。

(2) 简化网络内部节点的服务对象。采用聚集传输控制,具有相同 DSCP 的业务流组成一个宏流(macro-flow),核心路由器的服务对象是宏流而不是单流(micro-flow),单流信息只在网络边界节点保存和处理。

2. DiffServ 的业务分类

在 DiffServ 模型中,用户通过标记分组的 DS 字段来申请不同等级的服务。DS 字段定义为原 IPv4 包头的 TOS 字节或 IPv6 包头的流类型字节(traffic class octet)的前 6 位,称为区分服务编码点 DSCP,后两位保留,未在区分服务体系中定义。具有相同 DSCP 值的分组的集合称为行为聚合 BA。网络节点保留 DSCP 到每跳转发行为 PHB 的

映射。每个 DSCP 值只能对应一个 PHB，多个 DSCP 可能对应同一个 PHB。此处 PHB 指满足某一特定类别流量的转发需求的行为，如流量管制、流量整形、队列管理等 QoS 控制行为。

当分组进入路由器时，路由器根据分组携带的 DSCP 标记将其划归某一 BA，并选取与之对应的特定的 PHB 来转发该分组。具体而言，边界节点根据用户的流规定(profile)和资源预留信息将进入网络的单流分类、整形、聚合为不同的流聚集，这种聚集信息存储在每个 IP 包头的 DS(differentiated services)标记域(field)，称为 DS 标记(differentiated services code-point，DSCP)。如图 3-28 所示为 IP 包头的 DS 标记域。



图 3-28　IP 包头的区分服务标记域

目前，IETF 定义了以下几种标准的 PHB：

(1) 加速转发 PHB(EF PHB)[26]。它是一种"三低一保证"的业务，"三低一保证"是指低延迟、低抖动、低丢失率和带宽保证等四项指标。在任何情况下，从任何 DiffServ 节点发出的业务流的速率必须等于或大于设定的速率。EF PHB 在 DS 域内不能被重新标记。重新标记仅允许在边界节点进行，并且要求新的 DSCP 满足 EF PHB 的特性。当采用"隧道"技术时，外部分组也需标记为 EF。它的 DSCP 值(DSCP 是 DS 标记域中的具体值)是 101110，该业务是一种类似于高速的虚拟租用线 VLL 的服务。

(2) 确保转发 PHB (AF PHB)[27]。它是一种"预约带宽"的业务，在网络拥塞的情况下仍能保证该类业务流拥有一定量的预约带宽。当前定义了 4 类 AF，分别是 AF1、AF2、AF3、AF4。每一类 AF 业务的分组又可以被细分为三种不同的丢弃优先级。AF 的编码点 AF_{ij} 表示 AF 类为 $i(1\leqslant i\leqslant 4)$，丢失优先级为 $j(1\leqslant j\leqslant 3)$。运营商在提供 AF 服务时，为每类 AF 分配不同的服务资源。DSCP 值是 100xxx、011xxx、010xxx、001xxx。

(3) 选择型 PHB(CS PHB)。它是一种按"分级服务"的业务，它的 DSCP 定义值类型是 xxx000，它在 DiffServ 域中是按原 IntServ/RSVP 定义的不同级别进行转发处理。

(4) 尽力而为 PHB(BE PHB)。传统的 IP 分组传送服务，该服务只关注流量的可达性，对其他方面不做要求。任何路由器都必须支持 BE PHB。DSCP 值是 000000。4 类聚集流对应关系如表 3-5 所示。

表 3-5　PHB 与 DSCP 值对应的关系

DSCP	PHB	说　明
101110	EF	绝对 QoS
100xxx	AF4	介于 EF 和 CS 之间，每一类 AF 业务的分组又可以被细分为三种不同的丢弃优先级
011xxx	AF3	
010xxx	AF2	
001xxx	AF1	
xxx000	CS	兼容原 IntServ/RSVP 定义的不同级别，共 8 类
000000	BE	尽力而为

DiffServ 的结构模型把网络分成不同的域，只在域的边缘进行接纳控制，核心骨干路由只进行转发，不再保留业务流的状态信息。但是该模型不提供端到端的 QoS 保证，而将 QoS 限制在不同的域范围内实现，不同的域之间通过服务等级协议（SLA）进行协商。

3．移动环境下的 DiffServ

目前的 DiffServ 不能满足移动性要求，主要原因有两个：没有信令，不能做到实时控制；不能动态配置服务质量参数。无线环境下，需要对 DiffServ 进行以下扩展：

（1）增加信令协议，用户在移动终端和基站之间传送控制消息及相关参数，如移动终端的能量、当前的丢失率等。

（2）增加对移动性的支持，为移动主机预留带宽，或者赋予移动主机高优先级，使其在切换时能够抢占低优先级业务的带宽，或使用预留带宽来补偿无线链路的高误码率损失。并要求基站能够过滤掉部分不重要的信息，以解决有线和无线链路速率不匹配的问题，或减少终端的能耗。

因此，移动环境下的 QoS 策略，发展趋势是核心网采用 DiffServ，无线接入网既可以采用 IntServ，也可以采用 DiffServ，无线接入网内用信令协议支持动态资源分配，资源分配可以和移动主机位置管理信令相结合，以加快资源分配过程，减少信令开销。

4．DiffServ 中的端到端 QoS 实现——宽带代理

因为 DiffServ 还不能实现端到端的资源预留和接纳控制，于是出现了带宽代理 BB（bandwidth broker）。DiffServ 域之间通过 BB 进行 SLA 协商，使 DiffServ 能够实现端到端的接纳控制和 QoS 保障。当前 BB 的研究是实现端到端 QoS 管理的一个重要环节。

BB 实际上就是一个 DS 域内的资源管理器，它收集网络的拓扑和节点以及链路状态信息，管理网络资源，并结合策略服务器规定的策略进行接纳控制。BB 负责维护本地和周边其他区域签订的 SLA 协议，接受本地或其他 DS 区域的带宽分配请求，根据网络服务策略和当前的带宽资源分配情况来管理和分配带宽资源，配置边缘路由器。另外，它负责传递区域间的带宽分配请求消息，以此完成边缘节点的带宽预定。BB 的原型实现框架[27]如图 3-29 所示。

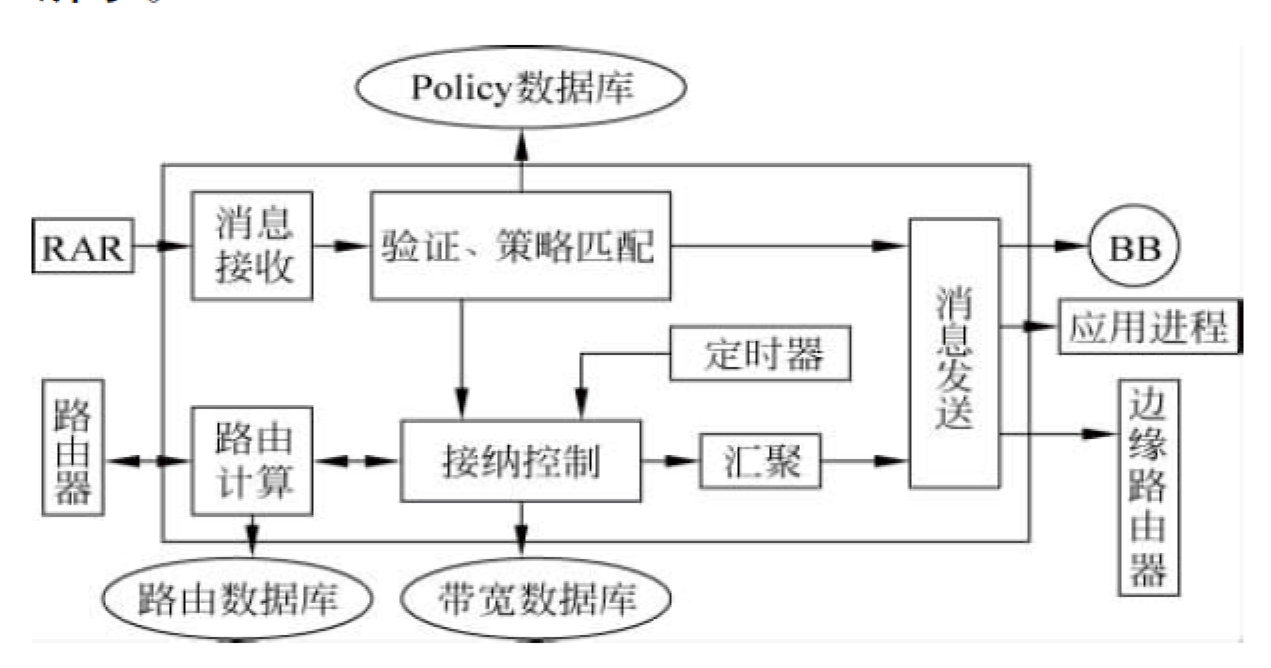


图 3-29　BB 的原型实现框架

其功能模块如下：

(1) 消息接收、发送模块。消息接收模块负责接收本地或其他 DS 区域的和资源分配有关的信息，主要包括资源分配请求消息及其应答消息。消息发送模块发送的消息包括三类：一是 BB 与其他 DS 区域 BB 之间传递的消息；二是 BB 向应用进程反馈的资源分配请求的结果，这类消息一般只有在端网络中才有；三是 BB 配置边缘路由器的消息。

(2) 验证、策略匹配模块。当 BB 的消息接收模块收到一个资源分配请求(RAR)后，首先要对消息的发送者身份进行合法性验证，并检查该请求是否符合网络服务策略，如服务等级、服务时间段、带宽要求有没有超出限定等。如果通不过，则直接由消息发送模块向请求者发出拒绝服务的消息，否则由接纳控制模块计算现有的网络资源能否满足请求。

(3) 接纳控制模块。接纳控制模块是 BB 的核心模块，它拥有区域内各路由设备的当前带宽资源使用情况。当有一个新的分配请求到来时，它利用路由计算模块获得业务流要经过的路由设备，计算在每个路由设备上是否有足够的带宽资源能满足分配请求。如果不能满足，BB 发出拒绝服务的消息。如果能满足且请求的目的地在本地，BB 对边缘路由器进行配置，并发出接受请求的消息。如果本地资源能满足要求但请求的目的地不在本地，则该类消息将被汇聚然后送往其下游 DS 区域的 BB。

(4) 汇聚模块。它对发往其他 DS 区域的带宽资源分配请求进行汇总，即根据所有的请求统计它们对各相邻区域的总的带宽请求，然后汇聚后的带宽请求被送到其他 BB。汇聚的主要目的是减轻 BB 的负担，以免过多的消息收、发和路由器配置行为。

(5) 路由计算模块。其作用是根据请求消息 RAR 中规定的流的源地址和目的地址，计算流在本地所要经过的入口路由器、内部路由器和出口路由器。

(6) 定时器。其作用是查看对一个用户的服务时间是否已经超过了 RAR 请求的服务时间期限。

基于 BB 的动态资源分配模型如图 3-30 所示。图中有两个端网络 Stub1、Stub2，它们通过区分服务区域 DS1 相连接。假设 Stub1 中的主机节点 A 要与 Stub2 中的主机节点 B 通信，它先进行带宽资源预留[14]。一个即时的带宽预留过程如下所述：

① Stub1 中的主机节点 A 向本地带宽代理 BB1 发出资源分配请求(RAR)。

② 假如 BB1 接受请求，它根据请求的目的地 B，向 DS1 的 BB2 发出请求。

③ 假如 BB2 接受请求，它把带宽分配请求发往 BB3。

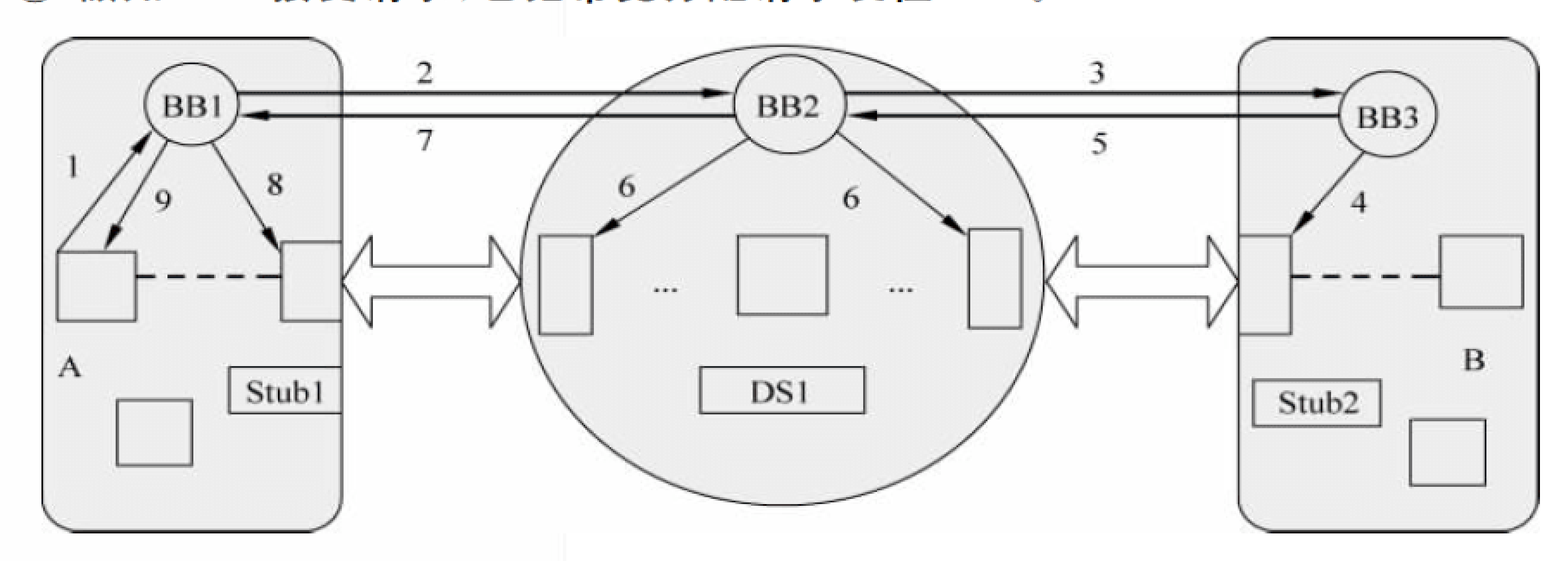


图 3-30 基于 BB 的动态资源分配模型

④ 假如 BB3 接受请求，它对数据流要进入的边缘路由器进行配置，更新本地资源信息。

⑤ BB3 向 BB2 返回接受请求的消息。

⑥ BB2 配置数据流要经过的入口路由器和出口路由器。

⑦ BB2 向 BB1 返回接受请求的消息。

⑧ BB1 配置数据流的出口路由器。

⑨ BB1 向主机节点 A 返回接受带宽分配请求的消息。

通过以上交互机制，主机节点 A 就能以预留的服务级别享受带宽服务。相反，如果某个 BB 发现本地资源不能满足要求，就向上游的 BB 发回拒绝服务的消息，直至该消息返回给主机节点 A。A 收到拒绝服务的消息后，可能降低服务级别或者另择时间发送请求。从而完成了边缘到边缘的带宽资源预定，实现了端到端 QoS 保障。

3.4.3 MPLS 多协议标签交换模型

传统的 IP 路由器分析每个数据包的网络层包头，并根据最长前缀匹配在路由表中搜寻下一跳。MPLS 最初是为了提高路由器的转发速度而提出的一个协议。MPLS 是属于三层的交换技术，引入了基于标记的机制，将选路和转发分开，由标记来规定一个分组通过网络的路径[29]。近年来，由于 MPLS 在流量工程和 VPN 服务中得到了很好地应用，该协议已经成为扩大 IP 网络规模的重要标准。现在也已经有很多的研究将 MPLS 与 IntServ 和 DiffServ 进行结合，用于提高 QoS 保障[30]。

1. MPLS 的基本原理

MPLS 是一种特殊的转发机制，它为进入网络中的 IP 数据分组分配标记，并通过对标记的交换来实现 IP 数据分组的转发。标记作为 IP 包头在网络中的替代品而存在，在网络内部 MPLS 在数据包所经过的路径沿途通过交换标记（而不是看 IP 包头）来实现转发；当数据包要退出 MPLS 网络时，数据包被解开封装，继续按照 IP 分组的路由方式到达目的地。

如图 3-31 所示，MPLS 网络包含一些基本元素，在网络边缘的节点即标记边缘路由器（LER），网络的核心节点即标记交换路由器（LSR）。由 MPLS LSR 构成的网络区域称为 MPLS 域。LER 节点在 MPLS 网络中完成的是 IP 包的进入和退出过程；LSR 节点

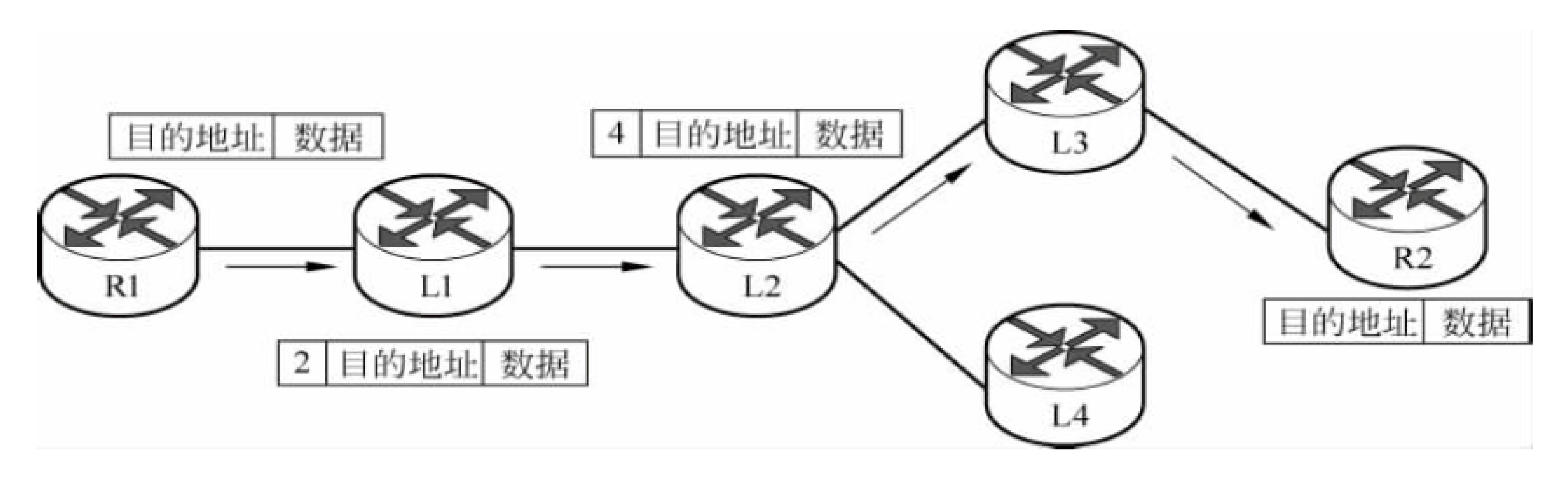


图 3-31 MPLS 的工作过程

在网络中提供高速交换功能。在 MPLS 节点之间的路径就叫做标记交换路径。一条 LSP(标记交换路由)可以看作是一条贯穿网络的单向隧道。MPLS 的工作流程可以分为几个方面,即网络的边缘行为、网络的中心行为以及如何建立标记交换路径。下面通过一个实例来说明 MPLS 的具体工作过程,图中 R1、R2 是普通的路由器,L1、L2、L3、L4 构成一个 MPLS 网络,L1、L3、L4 为 LER。

(1) 网络的边缘行为:当一个 IP 数据分组进入 MPLS 网络时,MPLS 第一次应用标记。首先,入口 LER(L1)分析 IP 包头的信息,根据目的地址的最长匹配,将该数据报进行流分类,并映射到合适的 FEC(转发等价类)上。简单地说,FEC 就是定义了一组沿着同一条路径、有相同处理过程的数据包。这就意味着所有 FEC 相同的包都可以映射到同一个标记中。对于每一个 FEC,LER 都建立一条独立的 LSP 穿过网络,到达目的地。数据包分配到一个 FEC 后,LER 就可以根据标记信息库(LIB)来为其生成一个标记。转发数据包时,LER 检查标记信息库中的 FEC,把相应的出口标签附加在分组的头部,如图 3-31 中将标签 2 加到 IP 包头,接着查找转发表,寻找出口标签为 2 的表项,找出相应的出口,把分组转发到该出口。

(2) 网络的核心行为:当一个带有标记的包到达 LSR(L2)的时候,LSR 提取入口标记,同时以它作为索引查找转发表。当 LSR 找到相关信息后,取出出口标签 4,并由出口标签 4 代替入口标签 2,从标记信息库中所描述的下一跳接口送出数据包。数据包到达了 MPLS 域的另一端,在这一点,LER(L3)剥去封装的标记,仍然按照 IP 包的路由方式将数据包继续传送到目的地。

(3) 标记交换路径建立:首先由 LDP(标记分布协议)和传统路由协议一起,在 LSR 中建立路由表和标签映射表。LER 接收 IP 分组,完成三层功能,并给 IP 分组加上标签,在 MPLS 出口的 LER 上,将分组中的标签去掉,继续转发;LSR 对分组不再进行任何 L3 处理,只是根据分组上的标签通过交换单元对其进行转发。

2. MPLS 信令的实现

目前 MPLS 实现信令的方式可分为两类,一类是 LDP/CR-LDP,是基于 ATM (asynchronous transfer mod)网络的。LDP 提供一套标准的信令机制,用于有效地实现标记的分配和转发功能,将网络层信息映射到数据链路层交换路由,并将每个 FEC 与一条 LSP 对应起来,建立标记交换所使用的一系列过程和消息。CR-LDP 是 LDP 的扩展,使用与 LDP 相同的信息和机制。另一类是 RSVP,它基于传统的 IP 网络。

作为建立、拆除、保护、重新路由以及重新建立标记交换路径的"信令",LDP 在 MPLS 体系中处于核心地位,是 MPLS 组网最重要的工具。LSR 周期性发送 Hello 消息表明它在网络中的存在。Hello 消息以 UDP 的形式发往普通的 LDP 端口。LSR 之间的 Hello 消息的互换使节点将链路和标记空间关联起来。当 LSR 决定与通过 Hello 消息发现的其他 LSR 建立普通的 LDP 会话时,LSR 将通过 TCP 端口发出普通的 LDP 初始化过程。双方通过传输地址可选参数来协商连接的传输地址,并根据地址值的大小来确定会话初始化过程中的那个 LSR 是主动节点。在成功建立 TCP 连接后,两个 LSR 就开始交换 LDP 初始化消息和协商会话参数。初始化过程由主动节点发起,该过程需要控制协商的重试次数。初始化成功后,两个 LSR 就称为"普通 LDP 对等实体",可以交换通告

信息。为了保护普通的LDP的正确操作，普通LDP通过TCP发送会话、通告和通知消息。

标签分发协议还包括RSVP。首先发送端发送PATH消息，除了普通PATH消息中所带有的TSpec消息外，该PATH消息还带有LABEL_REQUEST对象，用于各路由器分配标签；中间路由器接收到PATH消息，建立路径状态，并向下游转发PATH消息；接收端收到PATH消息后，向上游发出RSVP消息，该消息还带有LABEL对象，用于对各个路由器分配的标签进行分配及分发。中间路由器接收到RESV消息后，执行接入控制，并将LABEL对象中的标签值作为该会话的输出端口的标签值，同时分配新标签，并将新标签值与该会话的输入端口绑定。用RSVP分配标签的一个优点是使得沿着标签分配路径进行资源分配成为可能。

3. 移动环境下的MPLS

基本的MPLS并不支持移动性管理，移动MPLS将移动IP和MPLS结合起来，使MPLS网具备了支持IP移动性的能力。目前有两种移动MPLS技术：基本移动MPLS和分级移动MPLS(即H-MPLS)，下面重点介绍这两种技术[31,32]。

1）基本移动MPLS

在基本移动MPLS机制中，当移动终端MN进入非家乡子网时，发送一条注册消息给该子网的外地代理FA，FA通过正常的IP路由将该注册消息转发给MN的家乡代理HA，HA收到注册消息的同时也得到了MN的转交地址CoA。CoA可以是FA的IP地址，也可以是FA动态分配给MN的一个临时IP地址。然后HA向FA发送标签分发协议LDP的标签请求消息，FA则向HA回送LDP的标签映射消息，当标签映射消息到达HA后，HA和FA间的标签交换路径LSP就建立了。接着，HA通过查找标签表，找到以MN的家乡地址为转发等价类FEC的行，并把该行的出端口和出标签改为HA和FA间LSP所使用的端口和标签值。最后，HA通过已经建立的LSP发送注册应答消息给FA。FA收到注册应答消息后，在其标签表中增加一行，并把收到的标签值和端口值填到该行的入标签域和入端口域。

之后，若有通信节点CN向MN发送数据包，数据包会被路由到MN的家乡网，并被HA截获，这和移动IP协议中描述的一致。作为LSP入口的标签边缘路由器LER，HA会以收到的数据包中所包含的IP地址为索引，在标签表中查找该数据包的出标签和出端口。根据查找到的出标签和出端口，该数据包被加上相应的标签，并沿着HA和FA之间的LSP以标签交换的方式传送。FA收到数据包后将查找它的标签表（因为FA是该LSP的出口，所以标签表中相应的出标签和出端口为空），作为LSP出口LER的FA去掉数据包的标签，并把该数据包通过IP层送往MN。至此，MN就收到了通信节点发送给它的数据包。在基本移动MPLS中，每当MN移动到一个新的MPLS子网时，都要通过新子网的FA向其HA发送注册消息，并在新的FA和HA之间建立一条新的LSP。建立新LSP并将数据传送切换至新LSP所需的时间称为切换时延。如果LSP切换时延较大，通信节点和MN之间的数据通信就会中断，造成数据丢失。为减少切换时延，并保证切换过程中的通信，引入了分级移动MPLS。

基本移动MPLS只考虑了IP用户在一个较大范围内移动相对较慢的情况，当无线

接入网中因使用微蜂窝而造成 IP 用户在子网间频繁切换时，切换时延较大，容易引起通信中断。

2）分级移动 MPLS

分级移动 MPLS 克服了基本移动 MPLS 的缺点，具备微蜂窝移动性管理的能力。分级移动 MPLS 技术将一个 MPLS 网分为多个 MPLS 域，每个 MPLS 域中又有多个接入子网。

此外，分级移动 MPLS 还引入了一个新的概念，即外地域代理 FDA，每个 MPLS 域中有一个 FDA。MN 根据所接收到的代理广播消息来判断自己处于家乡网还是外地网。若处在外地网中，MN 就会向 FA 请求一个 COA，并发送注册请求消息给 FA。FA 把注册请求消息转发给本 MPLS 域的 FDA，而不是 MN 的家乡代理。如果 MN 是第一次移动到该 MPLS 域，FDA 就向 MN 的家乡代理转发注册消息。HA 得到注册消息并获得 FDA 的 IP 地址后，利用 LDP 向 FDA 发送标签请求消息。FDA 接到标签请求消息后向 HA 回送标签映射消息，并向 MN 当前所在 MPLS 域内子网的 FA 发送标签请求消息。标签映射消息到达 HA 后，FDA 与 HA 间的 LSP 就建立了。同样，FA 向 FDA 回送标签映射消息，当标签映射消息到达 FDA 后，FDA 和 FA 间的 LSP 也建立了。然后，HA 会在标签表中找到以 MN 家乡地址为 FEC 的行，并把出端口和出标签改为 FDA 和 HA 之间 LSP 的相应值。最后，HA 沿着其至 FDA 的 LSP 发送注册应答消息给 FDA，FDA 也会沿着 FDA 至 FA 的 LSP 转发该注册应答消息。FA 收到注册应答消息后，在它的标签表中加入新的一行，并在入标签域和入端口域中填入所收到注册应答消息的标签值和端口号。

与基本移动 MPLS 一样，在分级移动 MPLS 机制中，若通信节点向位于外地网的 MN 发送数据包，HA 就会截获这些数据包。在分级移动 MPLS 网中，HA 查找它的标签表，找出该数据包所对应的出标签和出端口。根据查找到的结果，数据包被 HA 加上标签，沿着 HA 和 FDA 间的 LSP 发送到 FDA。FDA 收到该数据包后会继续根据入标签值查找到相应的出标签值，即沿着 FDA 和 FA 之间的 LSP 把数据包转发到 FA。FA 收到数据包后，会查询它的标签表(因为 FA 是该 LSP 的出口，标签表中的出端口和出标签值都为空)。最后，FA 去掉数据包上的标签后，通过 IP 层把数据包转发给 MN，于是 MN 就收到通信节点发给它的数据。

若 MN 在同一个 MPLS 域中从一个子网切换到另一个子网，它将向新子网的 FA 请求新的 COA，并向新子网的 FA 发送注册请求消息。新子网的 FA 会把该注册请求消息转发给 FDA。收到注册请求消息后，FDA 向新子网的 FA 发送标签请求消息，然后新子网的 FA 向 FDA 发送标签映射消息，这样在新子网的 FA 和 FDA 之间就建立了一条新的 LSP，但是 HA 和 FDA 之间的 LSP 并没有改变。

显然，相对于基本移动 MPLS，分级移动 MPLS 不需要在 HA 和新子网的 FA 之间建立一条全新的 LSP，而只是在 FDA 和新子网的 FA 之间建立一条新的 LSP，从而大大减小了切换时延。切换过程中，MN 也可以通过发送绑定更新消息向原先的 FA 通知自己新的 COA，旧子网 FA 可以对 MN 的新绑定关系进行缓存。如果 FDA 使用过时的标签表向 MN 发送数据包，旧子网 FA 收到数据包后将与新的 FA 建立 LSP，并通过该 LSP 向新 FA 发送数据包，从而避免了切换过程中数据包被丢失。

在分级移动 MPLS 机制中，MN 在同一个 MPLS 域中的移动对其 HA 来说是透明的。这样一方面减小了切换时延，另一方面减少了整个网络中切换所需的消息交互，从而节省了网络资源。

3.4.4 异构网络的 QoS 映射

由于 QoS 的分层实现，不同层次上的 QoS 参数表达不同，要保证端到端的 QoS，需要 QoS 体系结构中的各层参数通过映射规则将用户的 QoS 需求忠实地从上层映射到下层，以便各层能够准确理解用户的 QoS 需求，更好地满足最终的 QoS。在协议栈不同层之间翻译 QoS 规格的过程叫 QoS 映射。

异构网络的 QoS 映射主要从两个方面进行研究，一方面是在同一网络中，不同层之间的 QoS 参数的传递与映射关系，以便在端到端跨越的每个网络中进行连接的建立或资源预留等；另一方面是不同类型网络上 QoS 业务分类和 QoS 参数指标之间的映射。下面分别进行讨论。

1. 同一网络、不同层间的 QoS 映射

根据网络的协议结构模型，通常将 QoS 参数表达为三层：应用层、传输层和网络层。要保证端到端的 QoS，就要 QoS 体系结构中的各层参数通过映射规则将用户的 QoS 需求准确地从上层翻译到下层，也就是不同层间的 QoS 映射问题。下面重点讨论上层协议与底层无线接入技术之间的映射，这也是异构无线网络与单一网络层间 QoS 映射的主要区别所在，其中底层无线接入技术与具体实现相关，包括 WLAN、WiMAX、3G 等。

1）异构无线网络体系结构

在提出异构无线网络体系结构之前，首先来介绍 ETSI 定义的宽带卫星多媒体网络（broadband satellite multimedia，BSM）的协议栈，如图 3-32（a）所示。BSM 协议栈的最

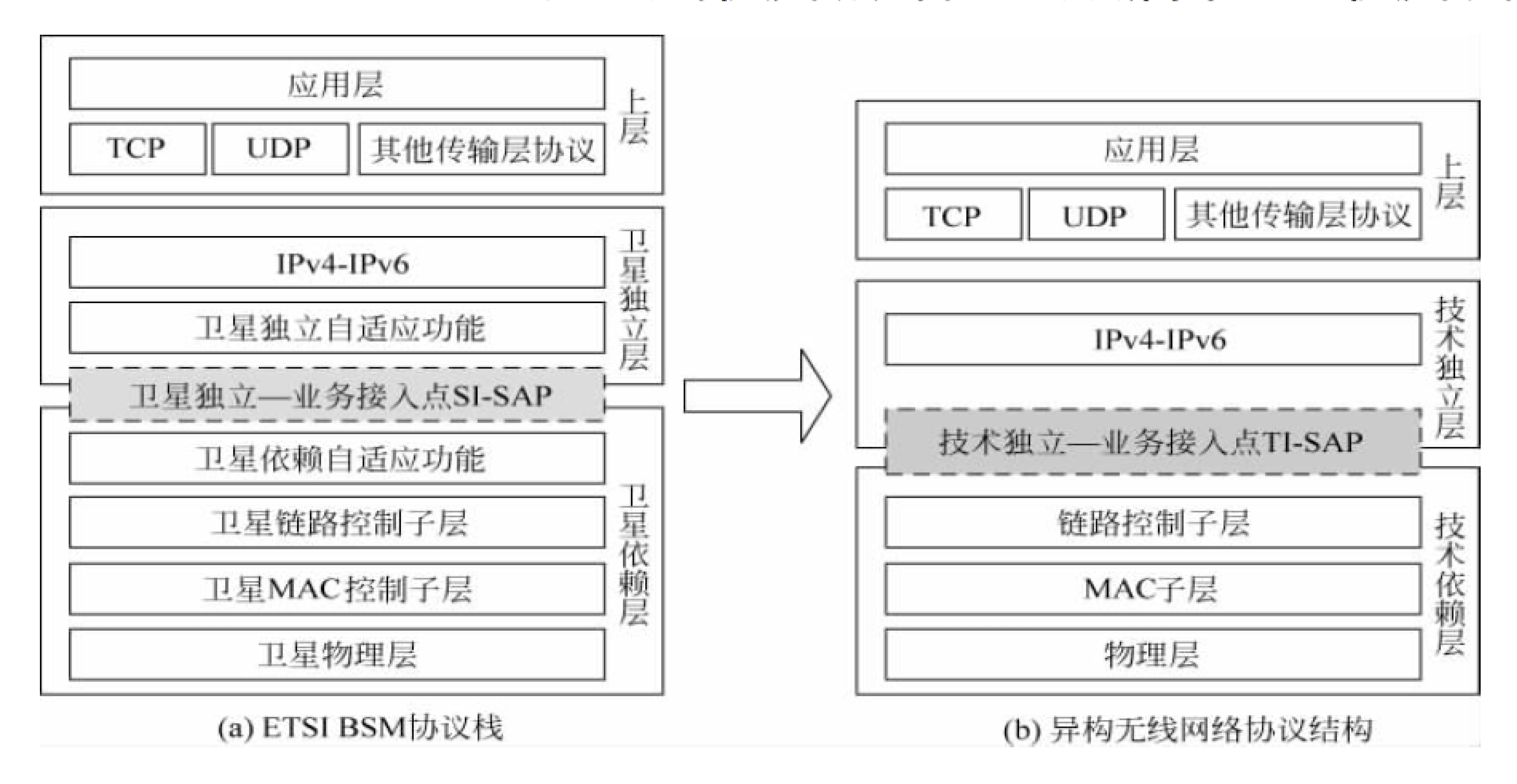


图 3-32 协议栈结构

上层是 TCP/IP 协议簇，与底层卫星接入技术无关，统称为卫星独立层（satellite independent，SI）。底层协议是严格的卫星技术依赖层 SD，包括卫星、链路控制层和 MAC 层，SD 通过卫星独立-业务接入点 SI-SAP 向 SI 层提供服务。这种体系栈结构除了被用做卫星通信研究外，还有其他更广泛的应用。

异构无线网络的体系结构可以借鉴 ETSI BSM 的协议栈结构，分为技术独立层 TI 和技术依赖层 TD，TD 通过 TI-SAP 接入点向 TI 层提供具有 QoS 需求的服务。这里技术依赖层涉及与具体实现有关的无线接入技术，包括 WLAN、WiMAX、3G 等。一般来说，层间 QoS 映射问题可以看成缓冲的映射。

2）技术独立层 TI 的 QoS 映射实施方案

IP 提供两种标记流量的方法：一种是由 IPv4 包头含有的“IP 源地址”“IP 目的地址”“协议”和 TCP/UDP 包头含有的“源端口”和“目的端口”组成的矢量。该矢量能够分类标记每个用户，但是不能分类标识具有同样性能需求的服务流。另一种是 ToS 域（IPv4 包头中占 8bit），前 6 位定义 DSCP 域。具有相同 DSCP 的分组处理方式。

综合服务采用第一种方式标识流量，基于 IntServ 模型的层间 QoS 映射模型如图 3-33 所示。其中资源请求通过 RSVP 对象传输，三组缓冲队列分别代表 GS、CLS 和 BE 服务。每个单独的队列传输具体的矢量流“IP 源地址”“IP 目的地址”“协议”“源端口”和“目的端口”。每个 IntServ 缓冲队列需要通过 TI-SAP 接口映射到 TD 层的缓冲区。

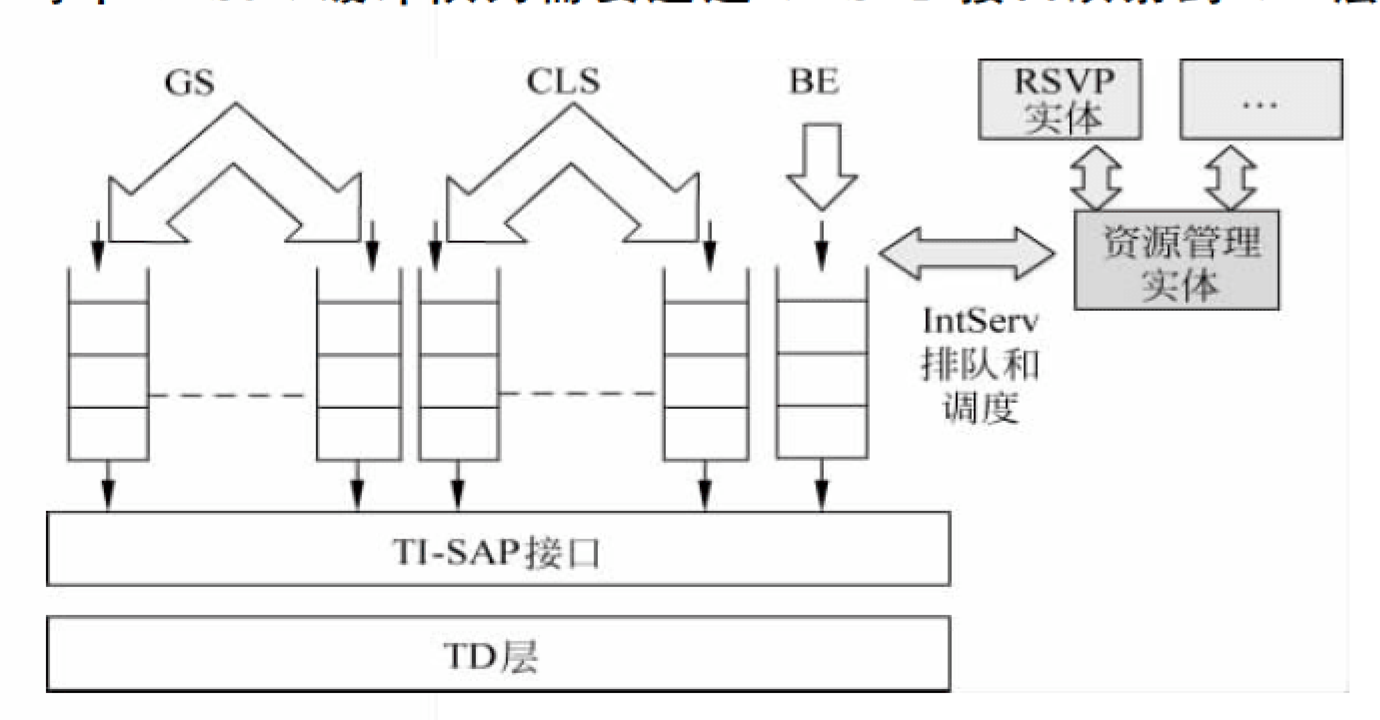


图 3-33　基于 IntServ 模型的层间 QoS 映射

DiffServ 模型采用第二种方案标识流，有关 DiffServ 的内容见上节。图 3-34 描写当 TI 层接收差分服务的层间 QoS 映射模型，服务流被区分为 4 组：EF（Expedited Forwarding）、AF（Assured Forwarding）、CS（Class Selector）、BE（Best-Effort）。DSCP 值唯一确定队列。

MPLS 通过 MPLS 头包含的 20bits 标签鉴别信息流。每个标签将关联控制模块，它将保证每个服务流中 SLS 特定的水平质量。MPLS TI 层采用的层间 QoS 模型如图 3-35 所示。每一个 SI 队列接收进入服务接入点的服务流。流和相关的队列会用标签项分类，标签从 1 到 n。

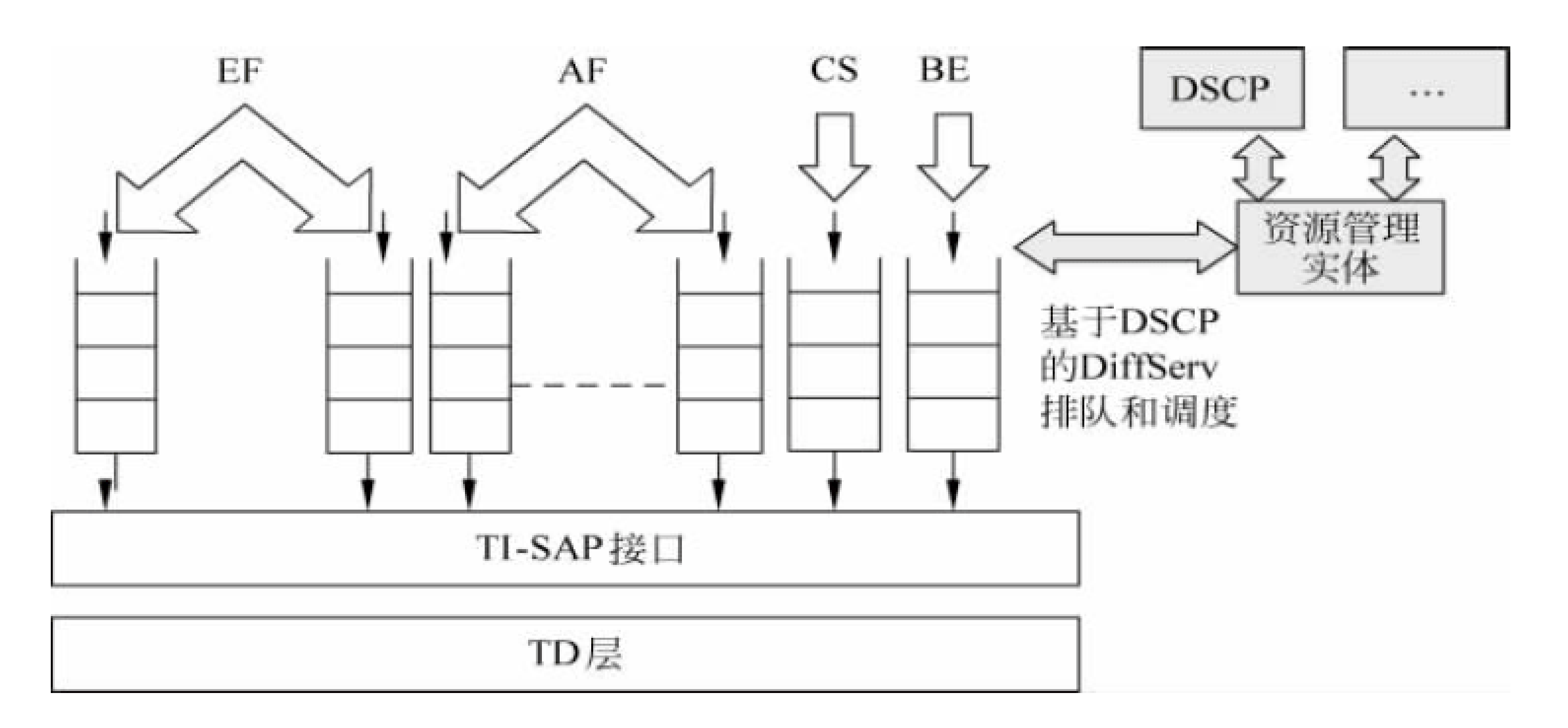


图 3-34　基于 DiffServ 模型的层间 QoS 映射

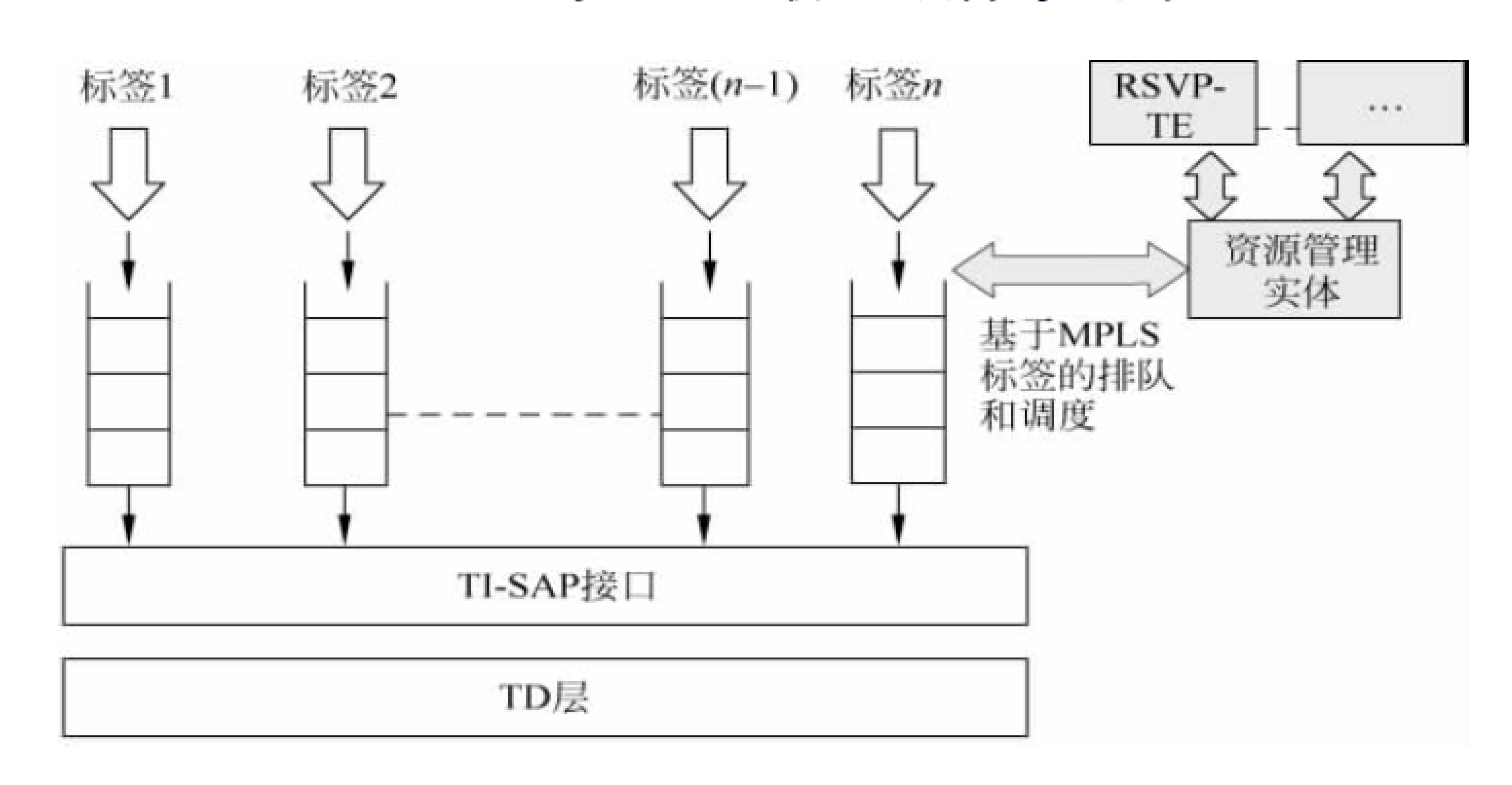


图 3-35　基于 MPLS 模型的层间 QoS 映射

2. 不同网络间的 QoS 映射

异构网络环境下，一个端到端的 QoS 会话可能要跨越不同的 RAT，如 UMTS、IP、WLAN 网络。但是每种类型的网络都定义了各自的基于 QoS 所支持的业务类型以及主要的具体的 QoS 参数或指标，以确保业务在各自网络上的服务质量。例如，UMTS 网络规范定义了 4 个等级的 QoS 保证，而 IEEE 802.11e 标准支持 8 个等级 QoS 保证。端到端的 QoS 业务需要涉及的所有网络能够依次进行 QoS 协商，能够以各自理解的方式处理端到端 QoS 需求，统一异构网络中的 QoS 参数。因此，不同类型网络上的 QoS 分类和 QoS 参数映射成为研究的重点。

现在将每种无线网络抽象成一个单独的自治系统 AS，异构无线网络间的 QoS 映射就等价于不同 AS 之间的映射，如图 3-36 所示。箭头处就是需要研究的重点，从一个网络向另一个网络服务质量需求的映射。

异构网络间的 QoS 映射最简单也最直接的方法是直接映射，以 TD-SCDMA 和 WiMAX 为例[33]，在图 3-37 所示的方案中，对于高带宽应用进行交互型与非实时轮询类别之间的映射，中等带宽应用进行交互型与尽力而为型之间的映射，对于低带宽应用进行后台型和尽力而为型之间的映射。直接映射方案的映射方法简单直接，但是只针对特定网络，不适用于重叠覆盖网络较多的情况，缺乏通用性。

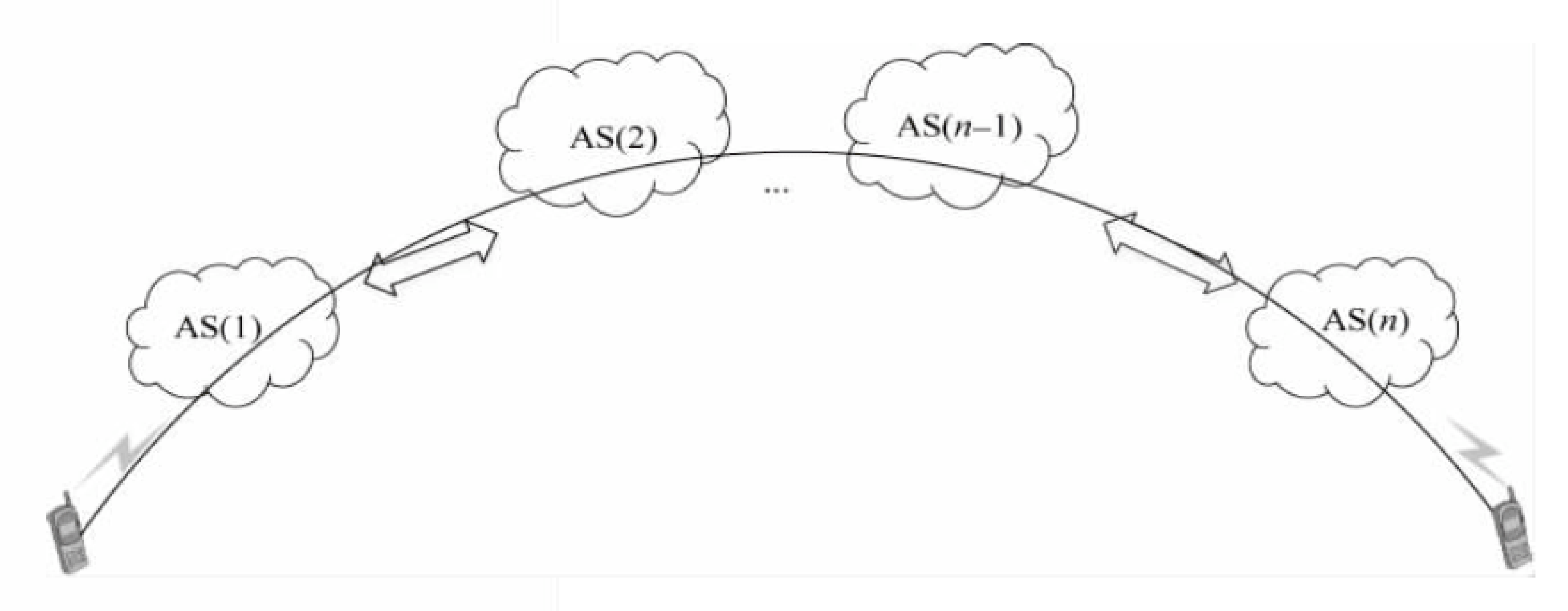


图 3-36 异构无线网络间的 QoS 映射

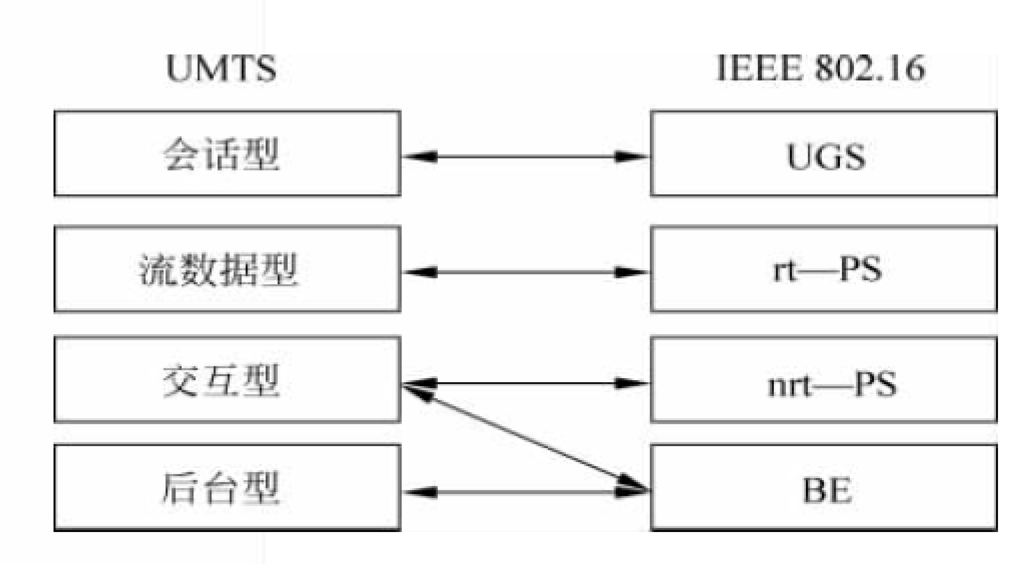


图 3-37 TD-SCDMA(UMTS)和 WiMAX(802.16)QoS 业务类别映射关系

通用的异构网络间 QoS 映射方案大体上可以分为两类。

第一种方案是指定一套划分粒度相对合适的 QoS 分类，将其他网络的不同 QoS 类别根据时延、时延抖动、丢包率等 QoS 特征参数与指定的 QoS 分类进行映射，从而以指定的 QoS 分类为桥梁，间接地在异构网络中跨越的不同网络类型之间进行 QoS 映射。

ITU-T 建议 Y.154[34]就是该种方案的一个典型例子。建议中将基于 IP 网络的分组传输性能参数分为 6 个 QoS 组。考虑的参数包括 IP 分组传输时延 IPTD、IP 分组时延抖动 IPDV、IP 分组丢包率 IPLR 和 IP 分组误码率 IPER。

IP QoS 分类准则可以实现不同网络技术之间的映射。但是基于 IP QoS 的分类映射还存在一定的问题。对于一些特别的业务，如实时视频分发、高带宽的 TCP 连接等，IP QoS 分类并不能够完全准确地反应 QoS 等级，需要根据具体业务新增一些分类。

另一种方案则是在已有的多种 QoS 分类的基础上，制定一套统一的 QoS 映射规则，通过映射规则，多样的 QoS 分类能够相互映射。例如参考文献[35]提出以重要的性能参数作为坐标轴构成应用业务图(application service map，ASM)，将现有的应用业务根据目标参数在坐标上进行表示。

图 3-38 中是一个 ASM 样例，分别以时延与丢包率为横纵坐标，构成应用业务图。从中可以看出，纵坐标丢包率共分为 4 个等级，横坐标时延划分为 16 个等级。ASM 图上的每个点的坐标范围为(0,0)～(3,15)。接下来将不同业务通过在坐标上的表示位置进行统一的映射，本图就是 UMTS QoS 分类在 ASM 上的映射。同理，可以将其他网络的 QoS 分类(包括 IP QoS)也通过 ASM 图进行统一映射。

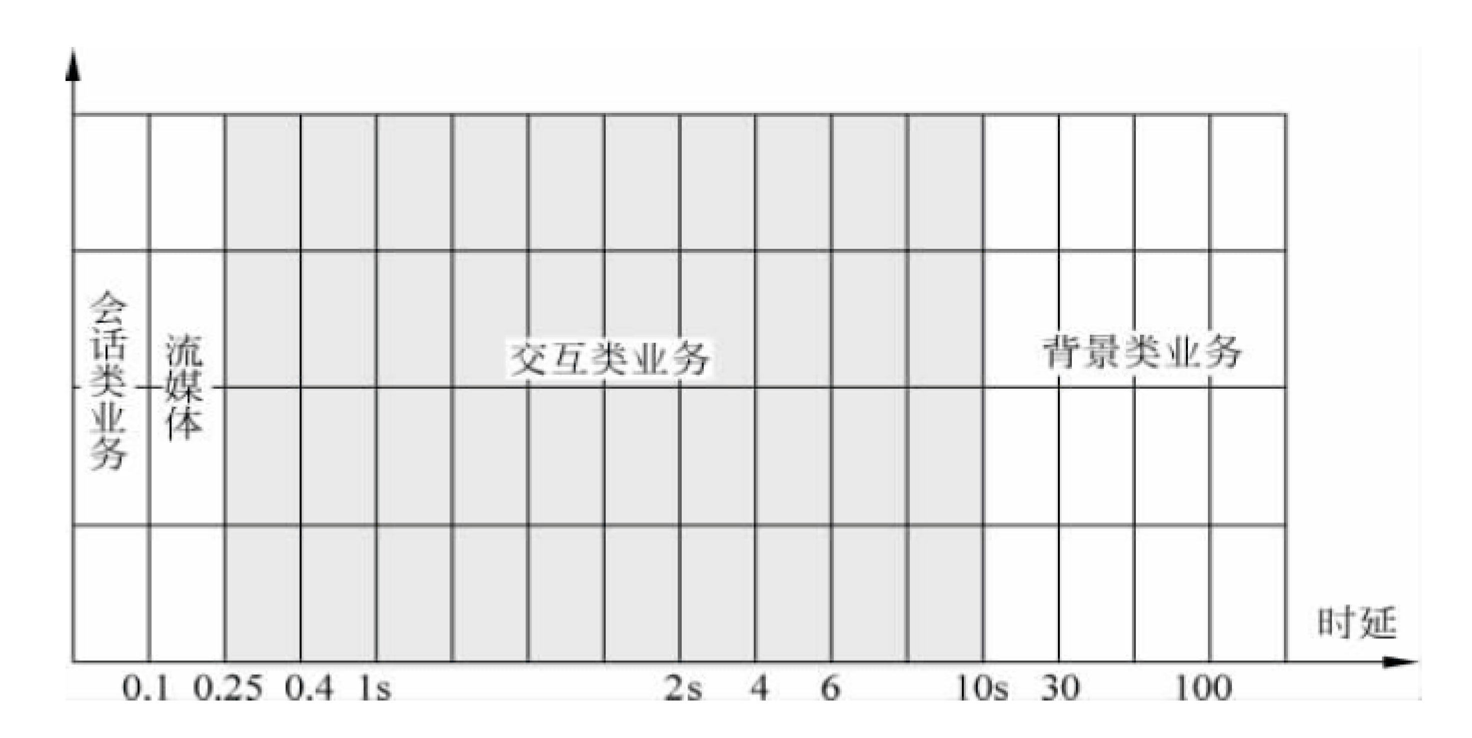


图 3-38 将 UMTS QoS 分类映射到 ASM 上

综上所述，方案一以现有的 QoS 业务分类为映射标准，能够将其他形式的 QoS 分类都映射到选取的 QoS 分类上，分类的细致程度较适中，但这种间接的映射方法难免因多次映射而造成误差的累加，增加了不准确性。方案二需要重新制定一套统一的 QoS 映射规则，映射的粒度相对适中，ASM 的方法表现直观，但只是从 QoS 的两个参数出发将分类进行映射，不够全面，局限性比较大，导致映射也比较模糊。

因此，在异构无线网络环境中，对 QoS 映射规则的划分就是在划分繁杂度与带宽利用率之间做一个权衡，划分粒度越粗，映射起来越简便，但对带宽的有效利用率越低，相反，划分粒度越细，可以更好地利用有限的带宽资源，但是映射起来越繁杂。所以能够在这两方面之间取得一个相对合理的平衡，粒度适中的 QoS 业务分类映射和具体的 QoS 参数在不同网络之间的映射还需要进一步的研究。

参考文献

[1] 陈山枝，时岩，胡博. 移动性管理理论与技术[M]. 北京：电子工业出版社，2007.

[2] 陈山枝，时岩，胡博. 移动性管理理论与技术的研究[J]. 通信学报，2007，28 (10)：123-133.

[3] Akyildiz I F，Mcnair J，Ho J S M，et al. Mobility management in next-generation wireless systems [J]. Proceedings of the IEEE，1999，87(8)：1347-1384.

[4] 蒋群艳. 全 IP 无线异构网络融合及其切换研究[D]. 上海：上海交通大学电子工程系，2007.

[5] 李明欣. 异构融合网络移动性管理的若干关键技术研究[D]. 北京：北京邮电大学网络技术研究院，2009.

[6] 华为科技. CDMA 标准的发展历程[J]. 电子科技，2000(23)：14-16.

[7] 唐晓梅，陈庶民. GSM MAP 协议处理分析[J]. 信息工程大学学报，1999(4)：5-7.

[8] Lin Y，Devries S K. PCS network signaling using SS7[J]. Personal Communications IEEE，1995，2 (3)：44-55.

[9] A. R. Modarressi，R. A. Skoog. Signaling System No. 7：a tutorial[J]. IEEE Communications Magazine，1990，28(7)：19-20.

[10] [ETSI/TC GSM 94] ETSI/TC，Mobile application Pare (MAP) specification，version 4. 8. 0. Tech. Rep. Recommendation GSM09. 02，1994.

[11] 张裕. 异构无线网络中无线资源管理若干问题的研究[D]. 上海：华东师范大学通信与信息系统

学院,2013.
[12] 李军,宋梅,宋俊德. 一种通用的 Beyond 3G Multi-Radio 接入架构[J]. 武汉大学学报:理学版,2005,51(2):95-98.
[13] 彭木根,孙卓,王文博. WiMax 与 3G LTE 网络互联与融合技术研究[J]. 电信科学,2007,23(1):10-15.
[14] Falowo O E,Chan H A. Fuzzy Logic Based Call Admission Control for Next Generation Wireless Networks[C]//In: International Symposium on Wireless Communication Systems. IEEE,2006:574-578.
[15] Suleiman K H, Chan H A, Dlodlo M E. Load balancing in the call admission control of heterogeneous wireless networks[C]//In: International Conference on Wireless Communications and Mobile Computing,Iwcmc 2006,Vancouver,British Columbia,Canada,July. 2006: 245-250.
[16] Falowo O E,Chan H A. Adaptive Bandwidth Management and Joint Call Admission Control to Enhance System Utilization and QoS in Heterogeneous Wireless Networks [M]//The governments of Europe/. Macmillan,1913: 222-237.
[17] Falowo O E, Chan H A. Dynamic Threshold-Based Joint Call Admission Control Scheme for Multi-Service Heterogeneous Cellular Networks[C]//In International Conference on Wireless Communications,NETWORKING and Mobile Computing. IEEE,2007: 3724-3727.
[18] Gazis V, Alonistioti N, Merakos L. Toward a generic always best connected capability in integrated WLAN/UMTS cellular mobile networks (and beyond) [J]. IEEE Wireless Communications,2005,12(3): 20-29.
[19] Chan H, Fan P, Cao Z. A utility-based network selection scheme for multiple services in heterogeneous networks[C]//In International Conference on Wireless Networks,Communications and Mobile Computing. 2005(2): 1175-1180.
[20] Yuksel M,Kalyanaraman S. Elasticity Considerations for Optimal Pricing of Networks[C]//In International Symposium on Computers & Communications. IEEE,2003(1): 163-168.
[21] Hal R. Intermediate microeconomics: a modern approach[M]//Workouts in intermediate microeconomics/. W. W. Norton,1987: 938-938.
[22] Braden R,Clark D,Shenker S. Integrated Services in the Internet Architecture: an Overview[J]. Ietf Rfc,1970.
[23] Wroclawski J. The Use of RSVP with IETF Integrated Services[J]. IEEE Commun,1996.
[24] Wroclawski J. Specification of the Controlled-Load Network Element Service[M]. RFC Editor,1997.
[25] Blake S,Black D S,Carlson M A,et al. An Architecture for Differentiated Services. RFC 2475 [J]. 1998.
[26] Jacobson V,Nichols K,Poduri K. An Expedited Forwarding PHB[J]. Ietf Rfc,1999.
[27] Heinanen J,Baker F,Weiss W,et al. Assured Forwarding PHB Group[J]. Ietf Rfc,1999,11(3): 82-89.
[28] 邱瑜,朱淼良. Diffserv 中带宽代理 BB 的实现[J]. 计算机科学,2003,30(4):47-50.
[29] Faucheur F L, Wu L, Davie B, et al. Multi-Protocol Label Switching (MPLS) Support of Differentiated Services[J]. TE) Management Information Base (MIB),RFC 3812,2002,15(4): 37-49.
[30] 贾世楼,尹歌峰. 基于 MPLS 技术的 IP QoS 的研究[J]. 通信技术,2003(9):44-46.
[31] 贺昕. 异构无线网络切换技术[M]. 北京:北京邮电大学出版社,2008.
[32] 董育萍,朱新宁,丁炜. 移动 MPLS 及其关键技术[J]. 现代电信科技,2003(8):14-18.

[33] 陈卓，余重秀，徐大雄. TD-SCDMA 与 WiMAX 系统的 QoS 类别分析及映射研究[J]. 现代电信科技，2006(6)：35-40.

[34] Babiarz B J，Chan K，Karagiannis G. Network Performance Objectives for IP-based Services，ITU-T Recommendation Y. 1541，February 2006. Babiarz06[J]. 2014.

[35] Mi S R，Park H S，Shin S C. QoS class mapping over heterogeneous networks using Application Service Map[C]//In International Conference on Networking，International Conference on Systems and International Conference on Mobile Communications and Learning Technologies，2006. Icn/icons/mcl. 2006：13.

第4章 网络可编程技术

由于目前网络架构对自组织、自配置与高效管理的需求，可编程网络成为网络发展的必然趋势。本章首先对可编程网络的发展历程、主要架构和实现方式进行概括，然后对可编程网络的代表框架软件——定义网络在无线网络与数据中心网络中的应用进行了详细的介绍，对相关技术的发展方向进行了探讨。

4.1 可编程网络

网络可编程性最初是指网络管理人员可以通过命令行对设备进行配置，后来有了可编程路由器、NetFPGA 等设备，这些设备的可编程性主要是对设备本身硬件电路级的可编程，即开发人员是通过编译代码直接控制这些硬件来实现自己的协议或者功能，这属于底层编程能力，可以类比于计算机中的汇编语言。开放"友好型"网络可编程接口，即可以快速容易实现并支持应用及各类部署方式的可编程接口，通过社区开发者的努力，使这些接口变得安全、健壮同时得到广泛的应用，最终成为正式标准，管理者可以通过这种高级的编程能力实现与网络设备的双向交互。通过软件管理网络，这种可编程能力类似于 Java 这样的高级语言，实现了更灵活、便捷的网络配置与管理方式。

4.1.1 可编程网络发展

1. 可编程网络的提出

广义而言，可编程网络表示可以通过基于软件的调控模块对其自身进行配置的网络，典型方式如通过应用程序接口（application programmer interface，API），软件层面上的功能协作允许网络的发展和软件研发周期的速率同步，这一速率明显优于迟缓的硬件研发周期。可编程应用包括及时的服务供应、灵活的资源管理、高效的资源共享以及对诸如云计算、物联网等新架构的支持。

传统的网络设备具有为系统或者网络提供配置的能力，但是这一

能力受到可编程性的限制。特别指出，传统设备大部分提供受限制的配置选项，可以想象为适合运营商需求所设置的“按钮”，而可编程性需要能够编程实现运营商所需的所有服务，例如，运行商需要自由定义可调节的新客户需求来满足应用程序和服务的市场定位，而不是被局限于设备供应商提供的配置选项，这时可编程设备明显比可配置设备更具有灵活性。

可编程网络不是一个完全的新概念，网络研发者早已意识到网络可编程性的缺乏，也提出过多种解决方式来解决这一不足。早在 1999 年 Campbell 等就在文章 A survey of programmable networks 中指出网络的可编程趋势并且预言高层网络可编程性将导致硬件从软件中分离，实现开放的网络接口，同时网络基础架构设备虚拟化以及网络服务的创新与研发周期缩短。这些预言可以作为软件定义网络(SDN)、标准化 API、网络虚拟化和云计算的展望。类似的，其他具有洞察力的文章，从早期 1996 年的版本，从可编程语言的角度讨论了网络可编程的设想，其协议与讨论旨在实现创建 Smalltalk of networking，这些思想均在 SDN 可编程环境下得到了复兴。

2. 可编程数据交换设备

网络架构中，无论交换机还是路由器，它们的主要工作都是实现数据信息在网络上的交换，这里的交换指完成数据信息从设备入端口到出端口的转发，因此，符合该定义的所有设备都可以被称为交换设备。由上面的介绍可知，网络可编程是指通过编程对设备进行配置，可编程交换设备(如可编程路由器、NetFPGA 等设备)可以对设备本身的硬件进行电路级的可编程，即通过编译代码直接控制这些硬件实现网络协议或者功能，这种可编程性虽然对于某台设备而言是底层编程能力，但是这些设备与网络高层编程能力双向交互，可以通过软件实现了更加方便灵活的网络管理方式。

这里以路由器为代表介绍传统交换设备，传统路由器的基本功能为路由和转发，其基本的架构和模块组成如图 4-1 所示。它由路由表控制转发决策，数据路径实现数据交换，即控制模块与数据交换模块紧密耦合在路由器设备中。

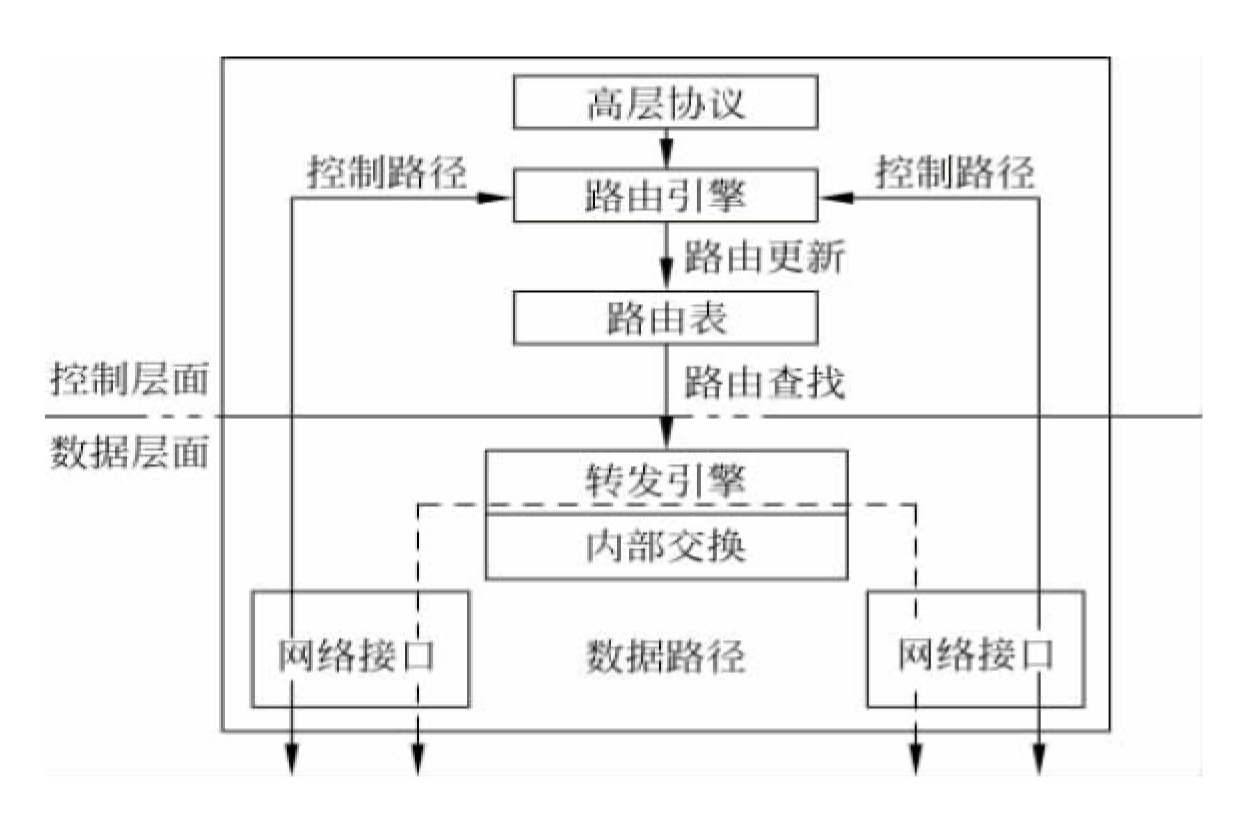


图 4-1 传统路由器基本框架图

与传统路由器相比，可编程路由器从功能上分解为若干组件和接口，并通过配置与编程，实现了灵活而动态地部署新服务、新协议，同时减轻了网络管理负担。可编程路由器除了具有传统路由的基本功能以外，其可编程性主要体现在以下几个方面[1]：

（1）软硬件的可编程性。可编程路由器的硬件架构允许用户重新定义功能，同时上层软件体系结构由功能划分清晰的模块或者 API 组成，允许用户重新组织这些模块或者调用接口来达到定制的目的。

（2）网络接口的可编程性。在高性能路由体系结构中，网络接口已由线卡（line card）所取代，线卡具有剥离包头，查找自身缓存中的路由表并进行转发的功能。线卡在转发数据之前，可编程路由器允许用户定制其他功能，如分类、整形、QoS 等。另外还允许用户识别自定义的数据包。

（3）数据路径的可编程性。可编程路由器应分离控制（管理）与数据（转发）两个层面，允许数据包根据用户自定义的条件进行数据路径的选择。

（4）上层协议的可编程性。用户可以重新定义或创建新的上层协议，以支持新服务与新应用。

（5）网络管理方式的可编程性。可编程路由器应该允许用户按自定义的格式组织脚本来对其配置，在监控和管理方式上，应支持多种或自定义的管理手段。

基于以上几点，近几年来，可编程路由器的研究取得了许多重要的成果。主要有软件架构方案、硬件架构方案以及数据路径可编程方案。

软件架构方案如 Click 可编程路由是一种在早期有影响力的软件路由，它将分类、排队、调度等简单的路由功能作为“元素”（elements）集合起来，组合为路由逻辑。尽管 Click 提供了快速的原型设计与调度的能力，并且为个人计算机上运行的软件路由提供了相当不错的性能，然而任何纯粹的基于软件的路由很难满足现代网络的性能需求。除此之外，Click 只针对数据转发层进行了可编程设计，没有与控制层的应用结合，不能进行动态配置。Click 路由器中的“元素”模型如图 4-2 所示。

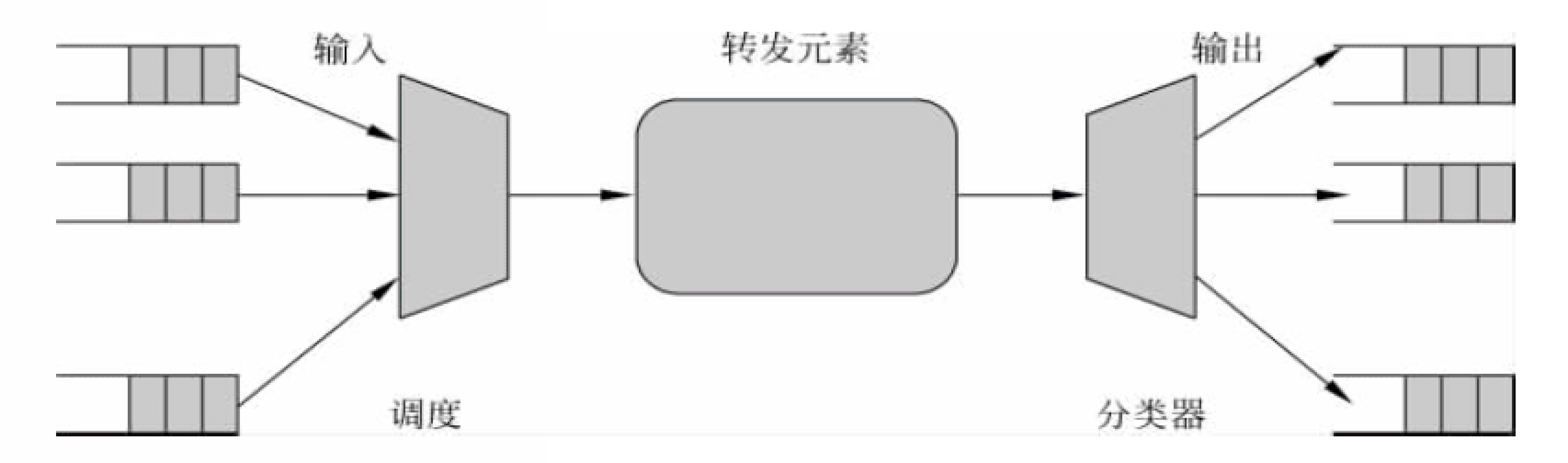


图 4-2　Click 路由器模型

硬件架构方案典型代表有 NetFPGA 方案，NetFPGA 是由美国斯坦福大学开发的低成本、可重用的硬件平台。其设计目标是开发出模块化程度高、开放性强、可重构的网络处理器，最大限度地减少网络研究的任务量。它的模块化设计理念，允许网络研究者在 NetFPGA 平台上设计实现千兆级别的高性能网络系统，允许将庞大复杂的网络处理器设计化整为零，自行设计或者调用已存在的硬件模块来组合实现。NetFPGA 平台由硬件转发层面和用户控制层面组成，平台框架如图 4-3[2] 所示。硬件转发层面是一个 PCI

板卡，包含两个FPGA、4个作为FPGA软核的千兆位以太网MAC控制器，内存由两片与FPGA核心逻辑同步运行于125MHz、字长36bit的SRAM和两片与FPGA异步运行于200MHz时钟频率、字长32bit的DRAM组成。用户控制层面主要包含架构管理模块和功能实现模块。架构管理模块主要包含用于对NetFPGA平台的硬件开发设计进行仿真和调试的计算机辅助设计(CAD)工具(ISE，ModelSim)和用于管理NetFPGA开发板与主机操作系统连接的板卡监管工具Monitor，主要实现板卡的系统配置、主机和NetFPGA数据的DMA传输、将开发板的片上寄存器与SRAM和DRAM映射到内存中，并通过对这些寄存器映像进行读写来控制NetFPGA的运行模式。功能实现模块主要包含：路由协议管理子模块，如ODPF路由软件Scone；系统测试子模块，如用于监听端口的Countdump计数器；基于Java GUI的用户管理界面。NetFPGA的优点在于，用户不需要大量电路设计知识，只需要少量的配置或者编程就可以实现定制功能的路由器，以达到转发自定义数据包或设计新网络架构原型的目的。但由于它仅是一块硬件转发卡，并非一款可以应用于大规模网络的可编程路由器产品，其功能和能力均还有限，并且它还存在一些缺点，如暂不支持IPv6，在时延、内存和带宽等方面还需要进一步改善等。

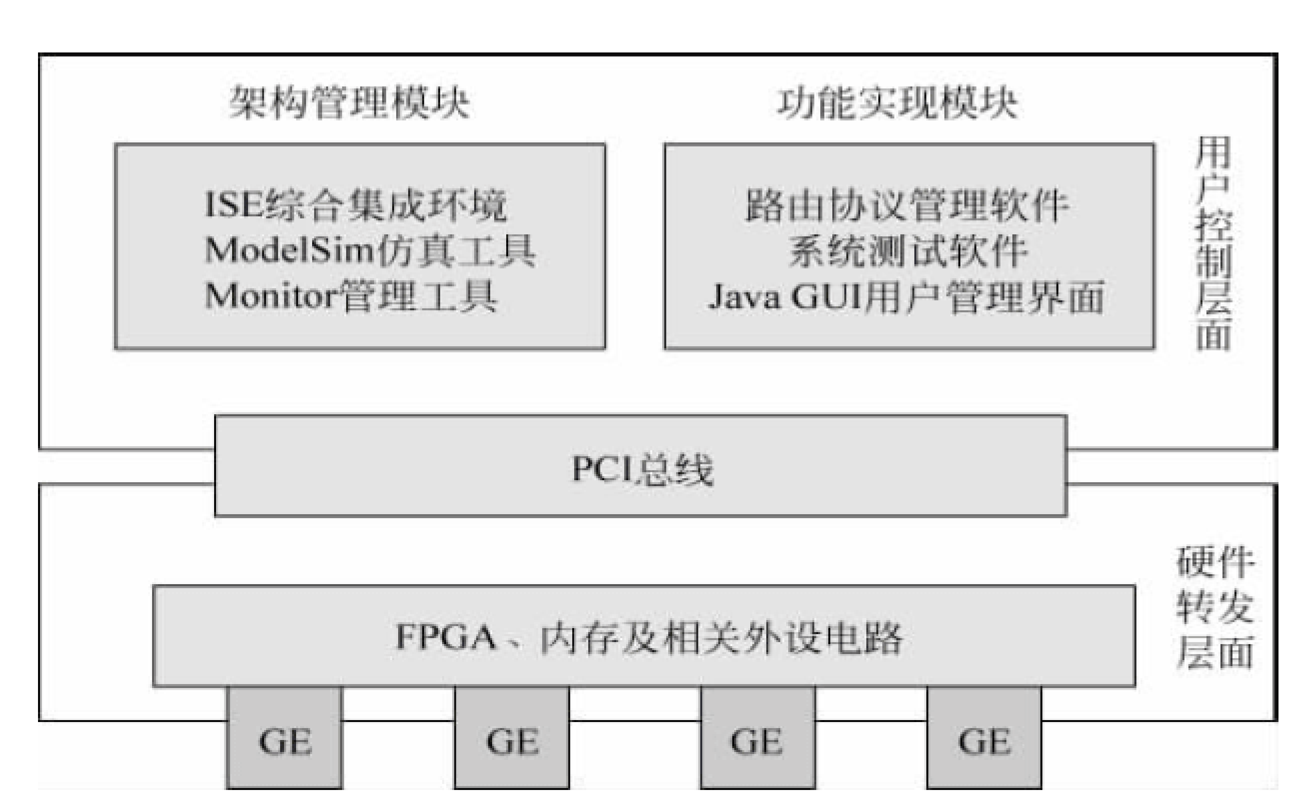


图4-3 NetFPGA平台框架(引自参考文献[2])

数据路径可编程方案指为了使可编程路由器达到组件和功能分解、数据与控制面分离和定制数据流路径选择的目的，研究人员在这一方面提出的一系列的标准和方案包括：

(1) 由IETF路由领域中关注开放可编程路由器体系结构和协议的工作组倡导的一种将路由器转发和控制层面分离的框架方案ForCES，其思想是把路由器分解为转发元素(forwarding elements，FE)和控制元素(control elements，CE)及连接二者的ForCES协议，从而使控制和转发层面所交换的信息标准化。

(2) 由斯坦福大学Clean Slate计划资助的一个开放式协议标准OpenFlow，是GENI的一个子项目，主要用于企业网络的控制和在现有网络上部署新协议。该技术主要包含三个部分：交换机上的数据流表、控制器和OpenFlow协议，控制器利用OpenFlow协议与交换机进行通信，将控制策略下发给数据流表，交换机根据数据流表对数据流进行处理，其结构如图4-4所示。厂商只需要在设备中支持OpenFlow协议而无须改动硬件架

构，用户则可以使用支持 OpenFlow 协议的网络设备建立异构网络或者创建网络测试床，该协议现已经被 Cisco、HP、Juniper 和 NEC 等许多厂商所支持。

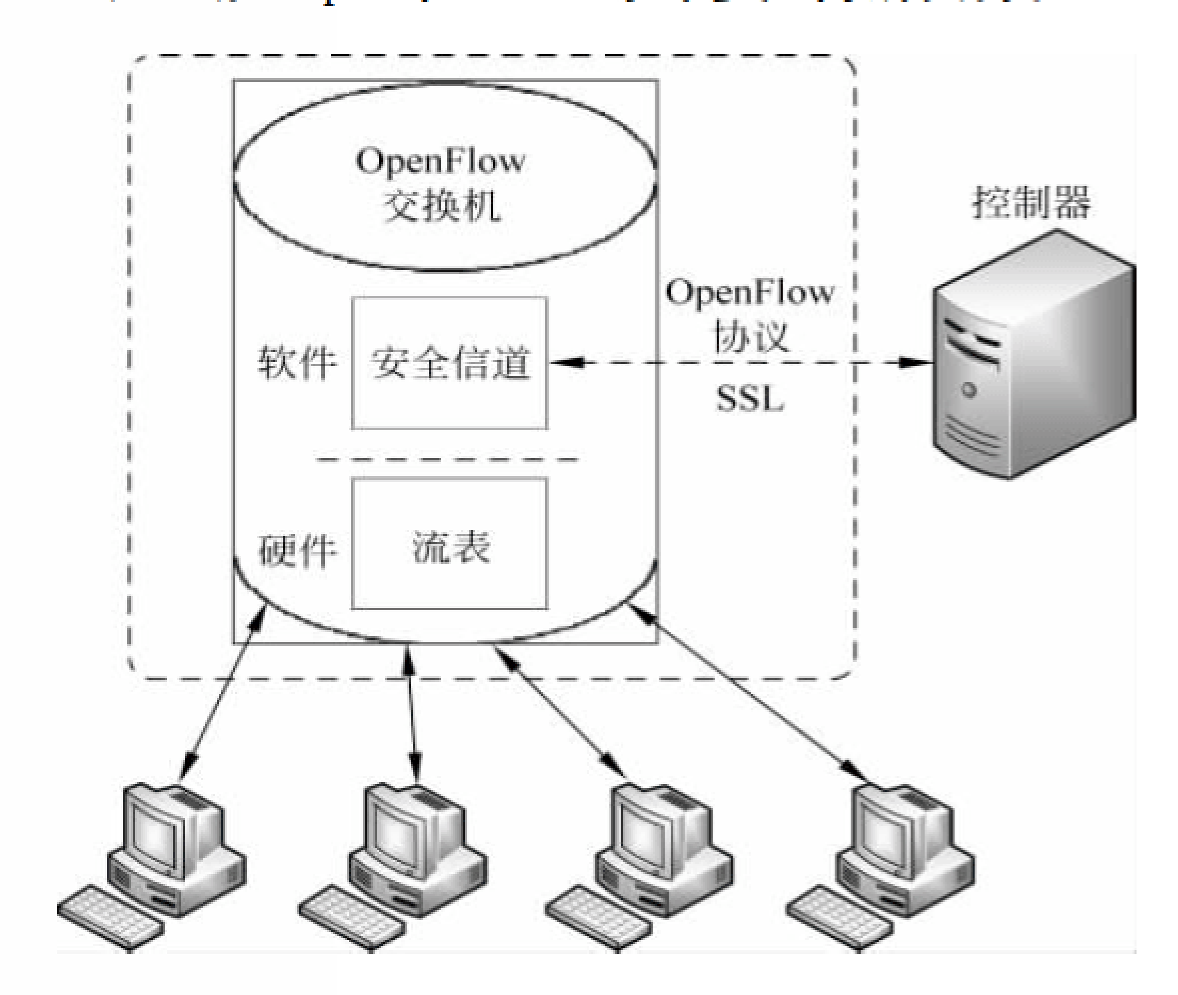


图 4-4　OpenFlow 结构

较传统路由器，可编程路由器具有模块划分清晰、配置动态灵活、编程组织轻松、服务部署迅速及应用前景广阔等诸多优点。然而可编程路由器的发展与创新还存在诸多待解决的问题：

（1）高性能、高转发速率的软硬件架构。到目前为止，仍没有一种转发速率在 10Gb 以上、成熟稳定的商用可编程路由器产品问世，而转发速率达百 Gb、甚至 Tb 级的高性能硬件架构更是亟待研究，这些设备需要操作系统、软件架构的创新。

（2）通用的配置与管理方案。一套通用的脚本配置语言和设备监测管理方案是必须的，这样才可以灵活而动态地部署服务和应用，减轻网络管理负担。

（3）统一的编程接口与规范。创建统一、标准的编程规范，使用户可以轻松地重新组织、定义路由器的功能，达到既满足大多数用户，也照顾到少数用户定制的需求。

（4）安全问题。由于可编程路由器具有重新定义功能、执行用户程序的能力，因此其安全问题是不容忽视的。但到目前为止，研究人员对此方面考虑不多，可编程路由器成熟而全面的安全保护机制仍有待研究。

4.1.2　可编程网络架构

1. 典型可编程网络架构

由于缺乏可编程复杂网络的创新，20 世纪 90 年代初期的时候就开始有研究者致力于创建可编程网络，当时主要有两个分支推进可编程网络：一组提出了 OpenSig 方法，另一组推进“主动网络”（active network）方法，但是他们的普遍共识是，可编程解决方法在于控制软件从硬件中分离出来并且开放控制和管理接口，用于创建可编程网络的网络

“块”(block)开始在各种形式的网络组件(例如 Click 模块化路由等)之后出现。最近,随着数据中心、虚拟化和云计算技术的出现,可编程网络的要求已经成为主流。下面将详细列举这些事态的发展。

1) OpenSig

20 世纪 90 年代中期,OpenSig 方法倡导在 ATM 网络中数据平面和控制平面的分离以及这两个平面之间信令的开放接口的应用。也就是在分离数据平面和控制平面架构中,采取开放的标准化接口,使得 ATM 交换机可以远程编程,从而更易于管理,OpenSig 社区积极致力于标准化这样的一个接口,并在各个科研院所建立了网络实验作品对这一思想进行探讨。在 Tempest 框架中,基于 OpenSig 理念,允许多个控制器平面同时控制 ATM 交换机的一个网络。然而,OpenSig 方式无法成为主流的主要原因是它定义的接口的静态特性。

2) 主动网络

主动网络的方法和 OpenSig 在同一时期流行,在 20 世纪 90 年代中期——互联网迅速商业化的时期,并且更多灵活控制的需求开始显现。主动网络方法旨在创造可编程网络,可以让网络快速创新,主动网络的研究推动主要是美国国防高级研究计划局(defense advanced research projects agency,DARPA)的努力,研究目的为解决在运行网络时快速部署新服务和实现动态分配的要求,OpenSig 网络的静态特性被认为无法支持这一需求。

主动网络的主要思想是主动控制网络节点,使得网络节点可以根据运营商的需要被编程来执行任意移动的代码,这种方式实现的价值在于它可以实现创新性的应用,充分利用网络的计算量,而且可以通过将服务从底层设备中解耦来加速研发进程。然而通过这种方式实现的灵活性,在另一方面也伴随着对其性能与安全的担忧。

主动网络方法包括两个编程模型:一个是胶囊模式,数据分组不仅包含交换的数据信息,还包括执行指令;另一个是可编程交换模式,其机制为利用在多样化的节点中执行代码。其中,胶囊模式为更激进的模式,明显不同于传统的网络运营模式,它也更能代表主动网络的思想。但是公平地说,这两种模式都为现代可编程网络提供了宝贵的可参考框架。具体而言,胶囊模式广泛引起研究者兴趣在于它可以升级数据平面处理方式,为整个网络传输路径提供一套简捷的数据交换方式。胶囊模式的使用为诸如主动式的网络负载均衡、多播、缓存等多种服务提供了支持,主动网络框架在 20 世纪 90 年代中后期在 DARPA 的资金帮助下被研究社区大力推行,各种有影响力的项目和著名的主动网络研究在此框架下展开。

主动网络范例是第一个对互联网结构重新设计的提议,现代出现的很多可编程网络概念,如控制平面和数据平面的分离、远程控制数据平面、虚拟化、网络接口等,都是萌芽于主动网络。然而,尽管主动网络突破了传统网络背景,它依然没能真正流行起来。主动网络范例失败的一个原因是在其已有框架内缺乏一个可信服的实用的网络应用程序用例,另一个原因是主动网络强调灵活地提供给端用户一个机会来对网络可编程,这一设想没有成为一个受欢迎的用例。

3) 虚拟化和云计算

虚拟化可以理解为这样一种技术:基于计算机科学,对原始资源提供一致性抽象以允许资源共享,最终实现实用目的。

虚拟化技术已经在目前大规模数据中心的背景下拥有巨大影响力，在虚拟化技术普及之前，不同关注点（例如安全、隔离、性能等）决定了服务器专用于特定的应用（例如专用的网络服务器、数据库服务器等）并且供应用于峰值负载。这导致总体资源利用率经常在10%～20%之间，这促使多个虚拟机实例抽象操作的使用，这些虚拟机可以创建在同一台物理主机上而又彼此之间互相隔离。这些虚拟机对端应用提供和底层的物理服务器相同的接口。随着虚拟机的克隆和移植（这允许将虚拟机以“快照”和“运输”的方式移植到任意一个当前未被充分利用的物理服务器）的可编程特性，物理资源可以被既安全又高效地分享。由于这些性能，虚拟化已经真正成为现代计算不可或缺的组成部分。

计算虚拟化在数据中心环境的普及催生了另外两个趋势：云计算和网络虚拟化。

云计算的主要观点是在一个虚拟化数据中心，通过网页编程的API提供服务，以此作为一个公用的计算方式。尽管效用计算早在1961年就由John McCarthy提出，但只是在近期才由云计算将其变成了现实。云计算模式与传统的数据中心的主要区别是在于灵活的虚拟化技术提供的服务以及对网页API的利用。编程服务的能力已经带来了很大的创新，而且通过对计算资源的可用率的均衡分配最大化计算能力。云计算的Holy Grail范例实现依据是安装一个通用的“网络结构”，该结构可以编程提供任意服务，不需要任何对网络节点的手动配置。这种基于结构的虚拟化数据中心的实施由于虚拟化网络的复杂性，已经被证明为一种难以实现的范例，以至于现在网络业界将数据中心的网络架构认为是其创新的瓶颈。随着传统垂直一体化网络设备，在支持云时代的应用的同时，通过运营商特定的命令行接口（CLI）手动配置网络交换设备带来一系列不良后果，典型表现为这个过程烦琐而容易出错。

随着每台机器可以配置数十台乃至上百台虚拟机的实现，在物理服务器内部实现的基于软件的管理程序切换接管了虚拟机内部的网络管理。需要指出的重要一点是，近来我们见证了意料之中的变化：物理端口被虚拟端口取代——这确实是网络发展史的重要节点，有着重大的结构性喻示。特别需要注意的是：使用管理程序辅以网络纤维结构组成的SDN技术和传统的经典端对端理论是功能对等的。另外，虚拟化/SDN混合结构同时包括了MPLS和中间盒的功能来完全分隔核心和边界。在这个新的结构中，基于SDN的网络纤维会成为新的核心，管理程序切换将成为新的边界。随后可以看到这些由管理切换（如Open vSwitch）组成的边界设备是软件定义的，因此是可编程的（使用如OpenFlow的协议）。这个设计范例从根本上转移到了软件控制，改变了创新速度，开启了无限可能。

尽管虚拟机对于某个特定的物理层服务器可以有着不受限制的应用，传统的网络虚拟化技术（如VLAN、VPN和覆盖网络）并不提供可以进行类比的虚拟化抽象技术用以对网络和物理层设施的去耦合。这里的VN抽象化应该像服务器端的VM抽象化一样，保证虚拟网络和物理设备的分离，同时隔离共享相同物理设备的占用，并为原网络提供相同的接口。在早期的OpenSig阶段，曾有过值得介绍的网络虚拟化工作：1999年提出的Genesis项目。这是一个虚拟化网络内核，能够按需生成虚拟网络结构。这里的“生成”是用在Genesis项目中，是操作系统领域用的术语，指同一台硬件机器上创建一个新的进程的过程。类比而言，生成网络意味着在相同的基础设施上创建新的网络结构。这个概念，尽管重要且新颖，和现代虚拟网络的虚拟抽象过程是不同的。正如虚拟机是一

个软件容器，封装了逻辑 CPU、内存、存储单元、网络等，并为应用提供了一个与物理机器相同的接口，虚拟网络也是一个软件容器，封装了逻辑网络组件，如路由器、交换机、防火墙等，这个软件容器表现为和物理层网络相同的应用接口。对于无线网络的 VN 抽象化在图 4-5 中有具体描述。这个抽象化对于 IT 管理者而言提供了极大的便利性，因为物理层网络可以以“纤维”的形式进行管理，提供可供按需使用、编程和重置的传输容量。

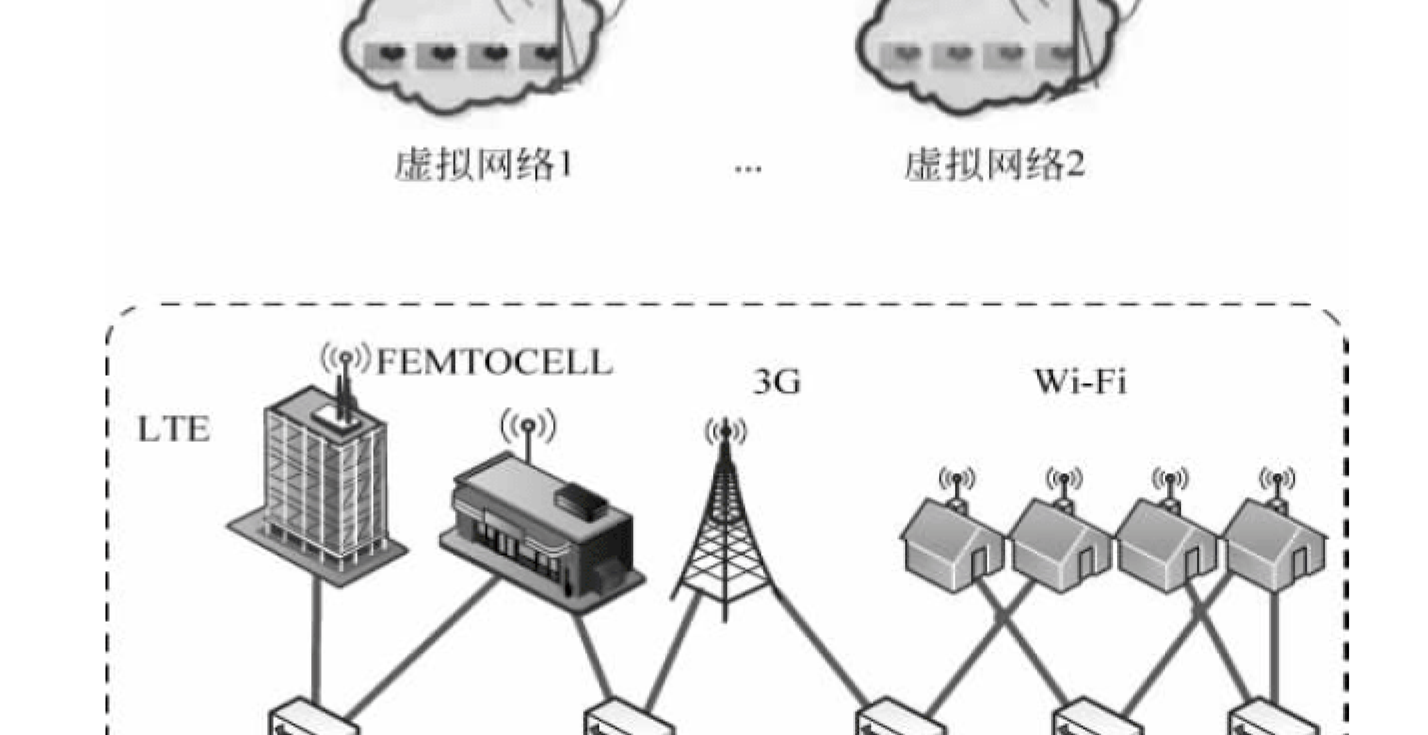


图 4-5　无线网络虚拟化

因特网社区中，当涉及通过覆盖化网络以最小的干扰产生创新型网络时，虚拟化技术也很受欢迎。覆盖化网络是一种建立在物理网络之上的虚拟逻辑网络，通过隧道和边缘设备互联。覆盖化网络和底层网络通过双地址空间进行去耦合。双地址空间是一种隧道封装，内部为虚拟地址，而外部为物理地址。覆盖化网络用于连接到本地网络的节点上，在相同的物理基础层可能会覆盖多层网络。每个覆盖层都是不同的高效逻辑网络，可以支持的服务类型包括如 L2 和 L3 随机策略，以及处理不同物理网络的接入控制表。这使得覆盖化网络在网络创新方面十分受欢迎，且无须干扰网络的核心部分。

有兴趣的读者可以参考网络虚拟化的相关综述性文献，以及云计算的相关文献来进行更详细的了解。

4）SDN

“软件定义网络”（SDN）的主要思想是允许横向集成系统通过允许分离的控制平面和数据平面，同时提供日益复杂的抽象集。SDN 已经通过网络可编程性彻底改变了互联网行业。SDN 将改善和简化网络管理和操作。学术领域已经证明，SDN 在很多工程项目试验中取得了巨大的成功。最近也发现了 SDN 在商业化部署方面具有较大的潜力。尽管 SDN 和主动网络模式的目的相同，即创建可编程的网络，但是它们两个的侧重点是不同的：在主动网络中重点在于数据平面编程，而 SDN 的重点是控制平面可编程，为了实现这一目的，开放性的接口是推进 SDN 技术发展的主要驱动力。将在 4.1.4 节结合可编程网络中的接口技术对其发展情况进行详细的介绍。

2. 代表性可编程网络项目总结

基于以上可编程网络架构，这里对具有代表性的相关项目进行整理，如表 4-1 所示，包括一些基于网络虚拟化与开放性接口项目。其中部分在 SDN 之前出现的基于转发与控制分离思想的网络架构称为“预 SDN”。

表 4-1　具有代表性的可编程网络项目

网络架构	项 目 名 称	时间	说　　明
主动网络	ANTS	1997	基于 Java 的主动网络工具包，作为麻省理工学院的 ActiveWare 项目的一部分被提出
	SwitchWare	1998	宾夕法尼亚大学的主动网络项目，主要集中于安全和性能问题
	CANEs	1998/9	全称为 Composable Active Network Elements，乔治亚理工大学研发项目
预 SDN	GSMP	2002	全称为 General Switch Management Protocol，用于标签交换
	ForCES	2004	在网络控制单元与转发单元之间定义了一个标准化接口
	RCP	2004	即 Router Control Platform，提出从路由器中分离出路由功能，并将其外包给一个单独的路由控制平面
	SoftRouter	2004	提出从数据包转发功能中将控制平面功能分离出来
	4D	2005	4D 即 Decision、Dissemination、Discovery、Data 4 个平面，将分布式的系统问题从设计逻辑中分割出来
	RaaS	2006	即 Route as a Service，提出提供由第三方供应商定制的路由计算作为一个服务
	PCE 架构	2006	即 Path-Computation-Element，PCE 作为一个应用放置于一个网络节点上或外网服务器上
	CogNet	2006	提出分离控制平面和数据平面，以一个可扩展的全局控制平面通过 API 控制分离出来的数据平面
	IRSCP	2006	即 Intelligent Route Service Control Point，允许外部路由通过外网智能地进行路由选择
	SANE	2006	通过一个“逻辑集中”的服务器实施企业网络安全/防护体系
OpenSig	Tempest	1998	旨在为云计算平台 OpenStack 提供集成测试的开源项目，基于 unittest2 和 nose 建立的灵活且易于扩展及维护的自动化测试框架
	Xblind	1996	哥伦比亚大学研发的用于创建宽带内核的工具包
	Mobiware	1998	哥伦比亚大学为移动多媒体网络研发的可编程 QoS 感知中间件
	NetScript	1999	用于数据包流处理的编程语言
	Virtual Switches	1996	作为 Xblind 项目的一部分提出了虚拟化 ATM 交换机
	Switchlets	1998	作为 Tempest 项目的一部分提出在 ATM 交换机上的动态可加载代码
	Virtual base stations	1998	作为 OpenSig 框架订阅项目 Mobiware 的一部分被提出
	Routelets	1999	在 Genesis 中提出的网络虚拟化

续表

网络架构	项 目 名 称	时间	说　　明
SDN	OpenFlow	2008	OpenFlow 定义了一个南向接口 API/协议标准,通过它可以让分离的专用控制器远程控制多个数据平面
	Floodlight API	2012	在控制器平面和 SDN 应用之间的一个 RESTful 北向 API
	Juniper APIs	2012	支持 JunOS SDK、XML、API(NETCONF)和 Contrail REST API
	OpenStack APIs	2012	OpenStack Neutron 是 OpenStack 的子系统,用于管理云环境中的网络
	FlowVisor	2009	基于 OpenFlow 的将网络以"切片"形式虚拟化的项目
	SecondNet	2012	提出一个虚拟数据中心抽象,多租户云的资源分配单元
	Network Virtualization Platform	2014	用于多租户数据中心产品

4.1.3 可编程网络接口技术

网络可编程性(network programmability)这一概念并不是新出现的,可以说它是一个可以存在于很多网络设备和软件组件中的特性。网络管理自从网络设备诞生时就产生了。传统方式中,管理器与代理采用相对松散的方式进行通信,操作时有很大延迟,个别情况甚至没有反馈机制。所以现在的问题在于,对于具体的网络设备,无论物理的还是虚拟的,不仅要进行管理,还需要研究设备之间的交互(interacted)。交互的目标是使得目标易于被编程操作,并能够与其他的软件进行双向通信,这样在组件间就形成了紧密耦合的反馈回路,这个概念与传统的网络管理模式有很大的不同。

为了实现新模式下的通信和交互,需要紧密耦合的双向可编程接口。这种接口需要能够快速而又容易地实现,才能够支持应用及各类部署方式。本节将对可编程接口进行详细探讨,并阐述如何将接口作为一种工具,以促进紧密耦合和双向通信信道的应用,并最终在控制器、所控制的网络设备和最终需要交互的应用程序之间形成反馈回路。

本节首先讨论了基于应用程序与网络分离思想的传统网络接口实现方式,如命令行界面(CLI)、NETCONF 与 SNMP,然后对 XMPP、JSON 等现代编程接口进行了介绍。

1. 应用程序与网络的分离

大多数路由器、交换器或防火墙这类现代网络组件直到最近才能够支持一小部分传统接口,以用于与这些组件进行通信。这类接口通常包括专用的命令行界面(CLI)、SNMP、CORBA,以及最近出现的某种形式的 NETCONF。这些语言通常都是很稳定的,需要设计并声明先验数据模型。事实上,这意味着编码通常就从这些接口中生成,而这些接口可作为内置在网络组件上的固件映像以及管理软件(或应用)。这意味着这些用于网络组件交流的接口必须被预先编程,而不是通过即时的学习来完成,同时语言中需要有用于定义消息结构和规则的语法以建立管理接口,并且这些协议通常使用二进制编码,这意味着它们在线路中难以被编程、调试或者以其他方式表述,导致这些语法模块在应用中对架构的适应性十分低下。

大部分实际情况下，并不对应用与网络组件的交流类型做出限制，并且应用服务也需要其中一种协议进行通信，更为普遍的情况是通过网元管理系统（element management system，EMS）进行网络管理通信。EMS 充当网络组件和应用之间的代理。实际上这便是所谓的应用程序—网络的分离（application-network divide）。然而，EMS 通常并不会以应用友好的形式来公开网络组件或其提供的服务，意味着对接口的编码和范式将会十分难以处理，并最终导致应用程序发出某一事件的信令后，需要等待很长时间才会发生。

1）命令行界面

命令行界面（command-line interface，CLI）是在用户图形界面得到普及之前使用的最为广泛的用户界面，所有的运营商必须要提供某种形式的命令行界面才能实现运营商与设备的通信。CLI 通常是基于 ASCII 字符的系统，其目的是为任意确定的设备提供默认和最低共同标准的管理接口，作为一个高效的语法分析器，CLI 可以根据字符串令牌进行动作，并能在输入命令之后尽快执行。大部分的设备都支持使用 Telnet 或者 Secure Shell 之类的公共协议对 CLI 进行远程访问，这些协议一般是跨网络运作，因此容易受到网络故障的影响，造成管理器与网络设备的通信受阻。出于稳健性考虑，大多数设备仍然会提供某种形式的硬件连接并支持本地交互的命令集。

通常设备制造商的命令行语法设定分为配置与查询两部分。在配置部分，经常有一个安全操作模式，使得管理器访问设备并改变设备的运行配置，有些设备允许运营商事先存有多个配置的备份，以实现用不同的方案进行配置，在某个特定的配置突然停止工作的情况下，系统可以回到另一个配置上继续工作。而查询模型允许管理器询问设备状态或者具体设备功能的状态，例如将系统的名字作为管理器可以查询的一个要素，以确保其可以配置在正确的设备上。

然而，一般情况下不同供应商所规定的 CLI 语法均不兼容。例如，有些系统允许大小写字母混合对系统进行命名，而另一个系统可能要求系统命名必须大写。这导致这些 CLI 缺乏实际操作上的语义兼容性，没有 CLI 的语法标准，导致推行受阻。在 CLI 范围内较早出现并且目前仍然广泛使用的网络可编程方式为使用 UNIX 脚本与设备的 CLI 交互，这一方式可以用 Perl、Expect 脚本、UNIX Shell 命令以及 Python 等多种工具实现，通过脚本编程，使其通过网络传输以及建立在 UDP/IP 或者 SSH 之上的 Telnet 会话协议连接到设备。然而，这种自动管理的方式存在一个明显的缺点，由于其对设备进行编程，收集数据然后调整配置或者采取某种措施，往往造成周期很长。并且对应用不够友好，尽管有些现代应用程序以 Perl 或者 Python 写成，但是这些应用一般情况下无法理解某个特定供应商的 CLI 语义语法。如果网络中同时存在多个网络供应商的设备，应用程序必须了解多种与设备交互的方法，包括设备的类型、厂商、型号以及固件镜像。大多数熟悉这种操作方式的人都认为这种方式需要应用开发者去了解本不需要了解的网络设备细节，这给他们带来了不合适也不必要的负担。

2）NETCONF

网络配置协议（network configuration protocol，NETCONF）由 RFC6241 定义，用以替代命令行界面（CLI）、简单网络管理协议（SNMP）以及其他专有配置机制。管理软件可以使用 NETCONF 协议将配置数据写入设备，也可从设备中检索数据。所有数据用

可扩展标记语言(extensible markup language,XML)编码,通过SSL或传输层安全这样安全、面向连接的协议,使用远程过程调用(remote procedure calls,RPCs)方式传输[3]。NETCONF由IETF制定,制定完成后于2006年12月发布。IETF在20世纪80年代后期开发了SNMP,直到今天,至少在统计监控(statistical monitoring)方面,它也是一个广泛使用的网络管理协议。在有了10年的SNMP部署经验之后,遗憾的是,实际情况依然与SNMP的开发初衷相违背,它并没有用以配置网络设备,而主要用于对网络监控。大约在2001年,IETF网络管理社区的成员与网络运营商共同讨论了这个情况。会议的结果在RFC3535中有记录,但总而言之,最终的讨论结果认为运营商主要使用专有的命令行界面(CLI)而不是SNMP来配置其设备。这次会议的另外一个重要结果是发现了这种行为背后的原因。这些原因的关键之处在于CLI具有很多运营商所喜欢的特征。例如,相比于基于BER编码(如二进制)的SNMP,CLI是基于文本的。另外,很多设备供应商并没有提供能够完全使用SNMP完成配置的设置。大多数信息需要事先从运营商处获得,而且只能进行只读操作。即使实现了完全的读—写功能,也只能用于自己的设备。虽然IETF制定了标准的MIB模型,但供应商仍然使用自己的MIB模型。因此通常导致供应商无法实现功能的扩展。最后,即便采用标准的MIB,在设计中某种实现下的语义最终不能与其他的相匹配,这使得SNMP更难(或者不可能)作为一种配置工具来使用。

正如前面所提到的,运营商通常倾向于使用脚本来管理他们的网络组件,但他们都发现CLI在许多方面都有所欠缺。其中最明显的便是结果的不可预知性(unpredictable)。结果的内容和格式容易发生不可预知的改变,甚至在不同版本的固件上的结果都不相同。某些改变会提供书面通知(written notification)或文档,有些改变不能提供这类信息,使得原本的困境雪上加霜。与此同时,Juniper网络已经采用基于XML的网络管理方式来与其远程设备进行通信(例如用于管理接口的协议),并且在其具体模型中,可使用本机语言(native language)操作CLI。这种新颖的方法作为一个更为统一的且对管理接口应用友好的方案,提交给了IETF,并在社区内广泛共享。RFC3535中记录了最早的提案和描述,最终导致了一种新网络管理协议的诞生,IETF称之为NETCONF。

简单地说,NETCONF提供了安装、操作和删除网络设备配置的机制。其操作运行在简单远程过程调用(remote procedure call)之上。NETCONF采用可扩展标记语言(XML)为其数据和消息报文进行数据编码。这些可以在TCP、HTTP或HTTPS等传输协议之上实现。通常情况下,NETCONF协议可以从概念上划分成4层,如图4-6所示。

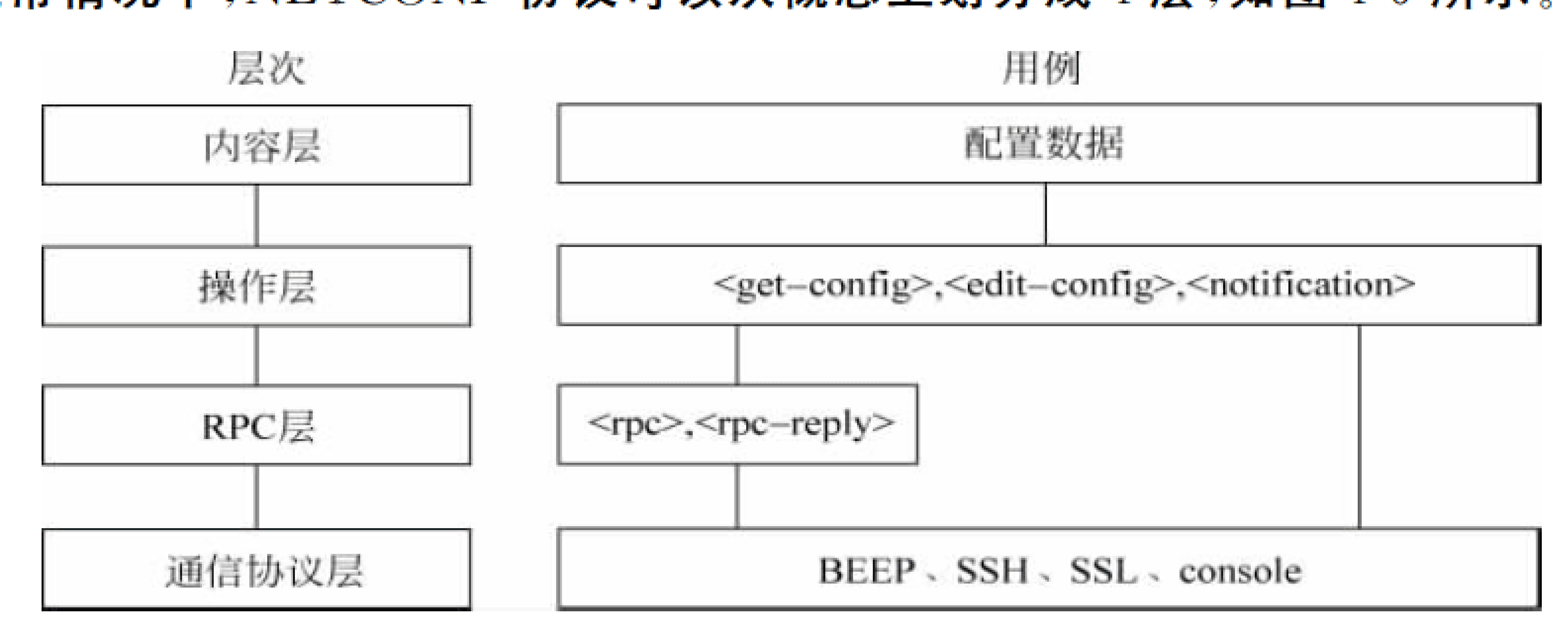


图4-6 NETCONF结构

NETCONF 的基本操作如表 4-2 所示。

表 4-2 NETCONF 基本操作

操作命令	操作功能
get-config	请求返回所有或一部分配置数据。传递的参数指定哪些配置数据需要返回，哪些具体元素需要获取。设备回复所请求的数据，如果设备不能满足请求，则返回 RPC 错误
Get	请求返回运行中配置数据和状态数据。该命令可以请求所有数据或指定一组元素
edit-config	修改配置数据。包含在命令中的操作指令对目标数据中的特定配置数据元素进行操作
Merge	编辑命令中携带的数据被合并到现有数据
Replace	编辑命令中携带的数据替换现有的数据
Create	创建指定的数据存储元素，命令中的数据插入。如果该元素已经存在，则设备返回 RPC 错误
Delete	删除指定的数据存储元素。如果元素不存在，则设备返回 RPC 错误
Remove	和 Delete 命令类似，但是如果元素不存在，则操作被忽略，并且不返回错误
copy-config	将一个数据存储复制到另一个。如果目标数据存储不存在，则创建它
Commit	将备选数据存储中的内容复制到运行中的数据存储。当设备功能不允许 copy-config 命令来修改运行中的数据存储时使用该命令
delete-config	删除指定的数据存储
lock 和 unlock	一台设备可能支持多个 NETCONF 与多个控制器会话，还可能继续支持其他配置机制，如 CLI 或 SNMP。lock 命令防止其他配置源干扰正在运行的一系列 NETCONF 操作。unlock 命令释放锁，并允许其他源在设备上操作。实际操作中，lock 命令应该只能在短时间执行
close-session	控制器软件通常在设备启动时就会打开一个 NETCONF 连接，并且只要控制器还在管理设备，就会一直维持连接。当控制器不再管理设备时，close-session 用来正常关闭连接

3）SNMP

简单网络管理协议（SNMP）是由 IETF 在多年前制定的，由一组网络管理的标准组成，包含一个应用层协议（application layer protocol）、数据库模型（database schema）和一组资源对象。该协议能够支持网络管理系统，用以监测连接到网络上的设备是否有任何引起管理上关注的情况。SNMP 在行业中已经存在了很长时间，多年来 SNMP 也经过了多次更迭得以改进。有三个版本的 SNMP 存在：V1、V2 和 V3。正如前面提到的，该协议目前仍然存在并被广泛使用，主要用于监控网络组件、组件的状态以及运行特性（performance characteristics）。如今，大多数应用网络正在使用或部分使用 SNMP 作为其组件管理策略，然而，其中大部分没有将其用于网络配置。

广义而言，SNMP 是一系列协议和规范，它们提供了一种从网络上的设备中收集网络管理信息的方法[4]。SNMP 管理的网络主要由三部分组成：被管理设备、SNMP 代理、网络管理系统（network management system，NMS），它们之间的关系如图 4-7 所示。

（1）被管理设备。网络中被管理的每一个设备都存在一个管理信息库（management information base，MIB）用于收集并储存管理信息。通过 SNMP 协议，NMS 能获取这些

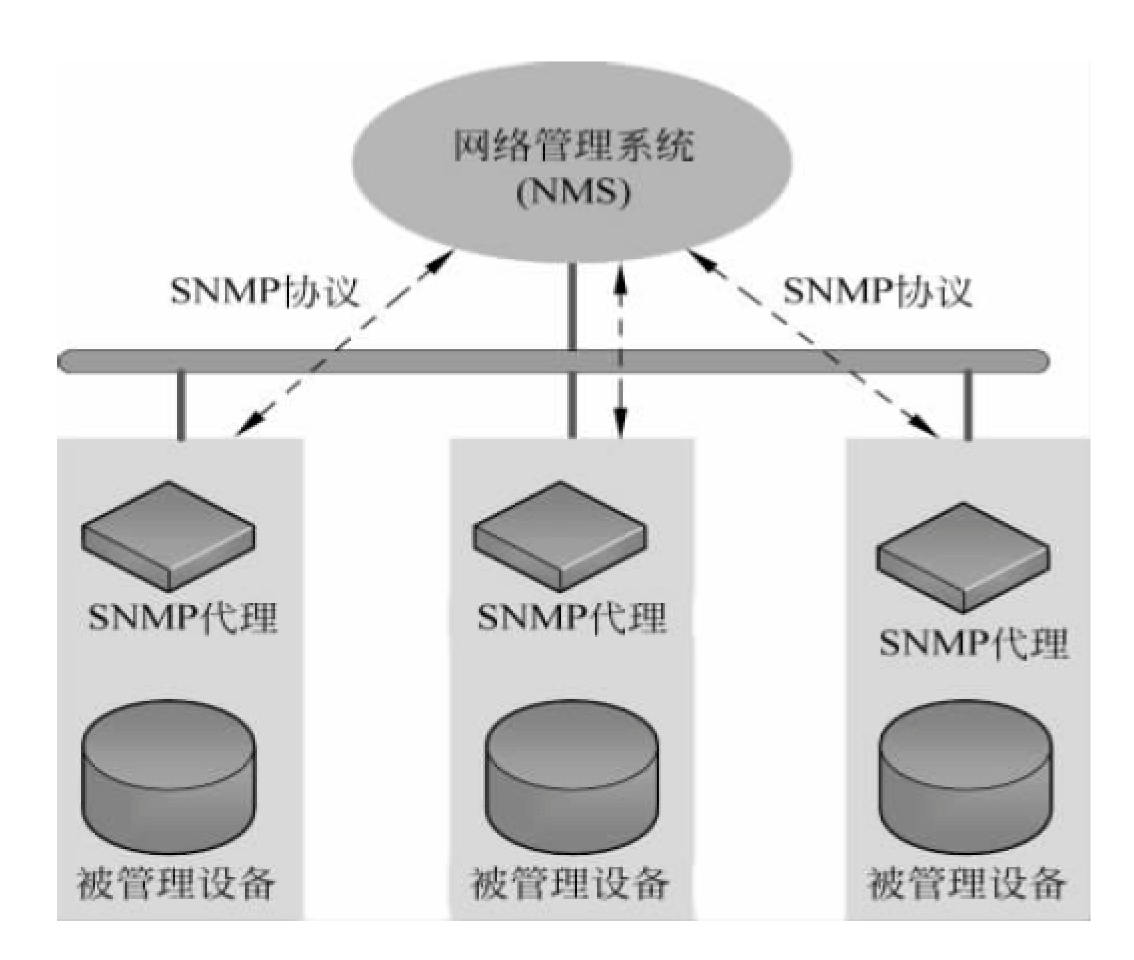


图 4-7 SNMP 管理的网络系统

信息。被管理设备，又称为网络单元或网络节点，可以是支持 SNMP 协议的路由器、交换机、服务器或者主机等。

（2）SNMP 代理。SNMP 代理是被管理设备上的一个网络管理软件模块，拥有本地设备的相关管理信息，并用于将它们转换成与 SNMP 兼容的格式，传递给 NMS。

（3）NMS。运行应用程序来实现监控被管理设备的功能。另外，NMS 还为网络管理提供大量的处理程序及必须的储存资源。

根据以上 SNMP 管理的网络模型，一套完整的 SNMP 系统主要包括管理信息库（MIB）、管理信息结构（SMI）及 SNMP 报文。

（1）管理信息库（MIB）。任何一个被管理的资源都表示成一个对象，称为被管理的对象。MIB 是被管理对象的集合。它定义了被管理对象的一系列属性：对象的名称、对象的访问权限和对象的数据类型等。每个 SNMP 设备（agent）都有自己的 MIB。MIB 也可以看作是 NMS（网络管理系统）和 agent 之间的沟通桥梁。它们之间的关系如图 4-8 所示。

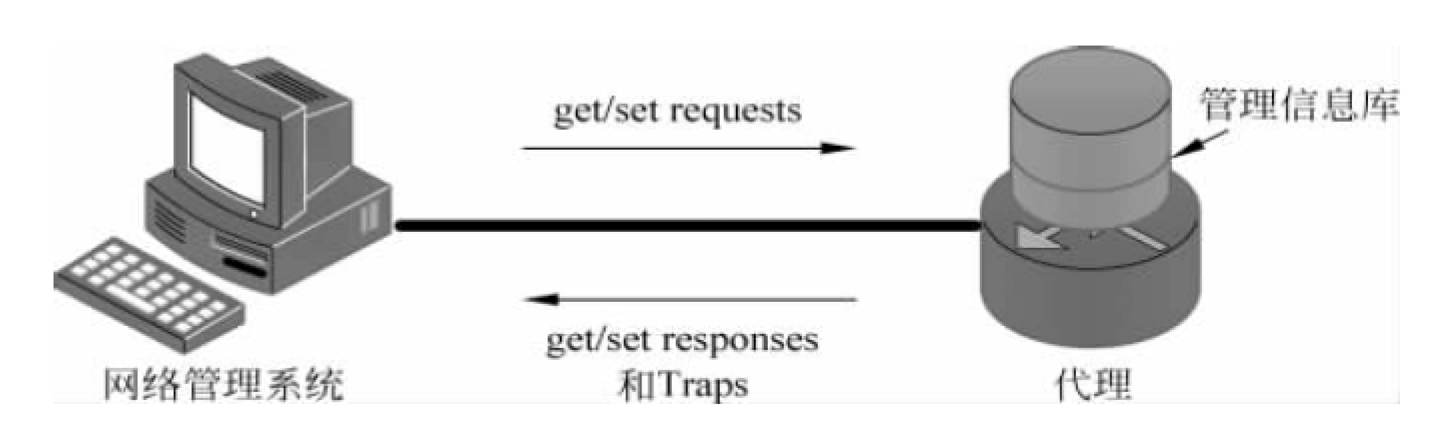


图 4-8 网络管理系统、代理与管理信息库关系

（2）管理信息结构（SMI）。SMI 定义了 SNMP 框架所用信息的组织、组成和标识，它还为描述 MIB 对象和描述协议怎样交换信息奠定了基础。

（3）SNMP 报文。SNMP 共有 5 种报文，所以其协议数据单元也有 5 种，用来进行管理进程和代理之间的交换，如表 4-3 所示。

表 4-3　SNMP 报文

报文操作名称	报文操作说明
get-request	从代理进程处提取一个或多个参数值
get-next-request	从代理进程处提取紧跟当前参数值的下一个参数值
set-request	设置代理进程的一个或多个参数值
get-response	返回的一个或多个参数值。这个操作是由代理进程发出的，它是前面三种操作的响应操作
Trap	代理进程主动发出的报文，通知管理进程有某些事情发生

2. 现代编程接口

通过上述分析，应用与网络分离的设计模式可以理解为网络管理通过应用程序与位于它和网络组件之间的代理或者翻译程序之间的通信来实现，但是这一方式的明显缺点是反馈太慢，对于要求实时完成的网络管理任务，这种方式的效率是明显不能满足的。在该方式基础上探讨现代管理接口及其概念，对于网络可编程接口，需要具备的关键特性有双向性、应用友好以及自我描述性。

订阅/发布模式定义了一种一对多的依赖关系，让多个订阅者对象同时监听某一个主题对象。这个主题对象在自身状态变化时，会通知所有订阅者对象，使它们能够自动更新自己的状态。该模式中，消息发送者向接收者发送消息，消息的发送者不直接将其消息发送给特定的接收者，而是将消息分成不同的类别，不需要知道什么样的订阅者订阅，订阅者可以随时提出需求。在这个模型中，订阅者可以表现出特定的“兴趣”，仅接收其感兴趣的部分，在这个过程中，订阅者不需要知道发布者的信息。这种情况出现了“消息总线”的概念，即将消息放在总线上，订阅者可以简单地接收信息，相比于状态管理需求较低的点对点系统，这种模式提供了更大的网络扩展性以及更动态的网络拓扑结构。消息总线可以是可靠的或者不可靠的，并且可以像虚拟的网络一样提供缓冲队列控制。

最早公开描述的发布/订阅系统是 Isis Toolkit 中的“新闻”子系统，描述在 1987 年的美国计算机协会(ACM)操作系统原理专题讨论会(1987 年 SOSP)的一份报告中，名为“分布式系统中的虚拟同步开发”。其中描述的发布/订阅技术是由 Frank Schmuck 发明的。

该模式的优点包括松耦合与扩展性，发布者与订阅者松耦合，甚至不需要知道它们的存在。因为关注的是主题，发布者和订阅者可以对系统拓扑结构保持一无所知。各自继续正常操作而无须顾及对方。在传统的紧耦合的客户端—服务器模式中，当服务器进程不运行时，客户端无法发送消息给服务器，服务器也无法在客户端不运行时接收消息。许多发布/订阅系统不但将发布者和订阅者从位置上解耦，还从时间上解耦发布者和订阅者。中间件分析师对发布/订阅使用的常用策略，是分摊出版者的份额来解决订阅者的线路阻塞(带宽限制的一种形式)。通过并行操作、消息缓存、基于树或基于网络的路由等技术，发布/订阅提供了比传统的客户端—服务器更好的扩展性。然而，在某些类型的紧耦合、高容量的企业环境中，随着系统规模上升到由上千台服务器组成的数据中心所共享的发布/订阅基础架构，现有的供应商系统经常失去这项好处；在这些高负载环境下，发布/订阅产品的扩展性是一个研究挑战。另外，脱离了企业环境，发布/订阅范式已

经证明了它的可扩展性远超过一个单一的数据中心，通过网络聚合协议如 RSS 和 Atom（标准）提供互联网范围内分发的消息。在交互时，为了能够即便是用低端 Web 服务器也能将消息播出到（可能）数以百万计的独立用户节点，这些聚合协议接受更高的延迟和无保障交付。

而该模式的缺点也是显而易见的，发布/订阅系统最严重的问题是其主要优点的副作用：发布者解耦订阅者。发布/订阅系统必须仔细设计，才能提供特定的应用程序可能需要的更强大的系统性能，例如有保证交付。发布/订阅系统的中介（broker）可能设计为在指定时间发送消息，随后便停止尝试发送，无论是否已收到所有用户成功接收消息的确认回复。这样设计的发布/订阅系统不能保证消息能够传递到所有需要这种有保证交付的应用程序。要完成有保证交付，这种发布者/订阅者在设计上的紧密耦合，必须强行脱离发布/订阅架构。发布/订阅系统中的发布者会“假定”订阅者正在监听，而实际上可能没有。在有少量发布者和订阅节点的小型网络和低信息量时发布/订阅能够自如伸缩。然而，随着节点和消息量的增长，不稳定性随之增长，限制了发布/订阅网络的最大可扩展性。大规模时吞吐量不稳定的例子有负载激增、速度变慢和 IP 广播风暴等。在使用中介（服务器）的发布/订阅系统中，同意中介发送消息给大量订阅者，会引发安全问题，中介可能被“欺骗”，从而发送错误的通知，即使不依赖中介，订阅者也有可能收到未授权的数据。

1）XMPP

XMPP（the extensible messaging and presence protocol）是发布/订阅模式的一个例子，并且在很多发布/订阅系统中都有应用，它是开放协议和内核代码的即时消息处理系统（Jabber IM）的基准协议技术，提供了一种开放式的、基于 XML 的、能在分布式网络中传输即时消息和在线发现的标准，并解决了不同 IM（instant messaging）系统间互操作的问题[5]。XMPP 在设计上沿袭了 Internet 上最成功的消息系统，即 E-mail。其路由处理的内核采用国际惯常的逻辑寻址机制，代表格式为节点@域/资源。在 XMPP 中，这种模式被称为 Jabber ID（JID）。其中，域可在 DNS 中查找，类似于电子邮件地址中的域名部分；节点可表示某 IM 用户、一类应用或某项服务；资源为一类连接标，能让某单一用户多次重复登录连接。每个用户都有自己的本地服务器（即自己的注册服务器），并从该服务器上接收信息，所有从一个客户端发给另一个客户端的消息和数据都必须通过服务端。每一个 XMPP 服务器都独立于其他 XMPP 服务器，并且拥有其自身的用户列表，通过 Internet，这些服务器构成了一个类似 E-mail 系统的分布式网络。服务器知道一个用户什么时候在线，这个能力被称为在线，也是即时通信的核心所在。XMPP 通过两个重要特性提供这些 IM 标准功能：首先是一个允许消息系统间协同作业的开放协议；其次是建立在 XML 上的强大根本，它使得非但是两个人之间的通信，甚至是应用软件之间的通信成为了可能。XML 是 XMPP 信息传输的核心，它最重要的作用是系统的底层可扩展性，它能表述几乎任何一种结构化数据。XMPP 是由 Jabber 公司创立、用于现场消息路由处理的 XML 数据流协议，是即时消息处理系统的基准协议技术，可以为网络间连接提供安全和易于实现的编程语言环境。

XMPP 的架构非常分散，任何人都可以运行自己的 XMPP 服务器，没有一个需要所有人连接的中心主服务器。发布者和订阅人的私人团体也能够搭建 XMPP 服务器，所有

的发布者和订阅者都采用基于主题的方式在服务器上注册，只有这样才可以在多方参与的会话中筛选出有效的会话。服务器能够支持多重会话，如许多即时消息服务器能够同时支持用户组间的多个私有多路径的会话。

XMPP 作为一种 IM 通用协议，在应用领域主要以 Google Talk、Jive Messager 为代表，在 XML 的结构化数据中扮演通用的"传输层"的角色，可以实现让数据高效路由到最合适的请求源，除了实现 IM 的互操作性，还可以通过各类应用实现现场的实时信息处理，如 CRM、协同软件等都可以作为 XMPP 的客户端，所有实体对于 XMPP 服务器都是透明的，实体中任何用户的状态对于其他用户都是可见的。

2）JSON

JSON(JavaScript object notation)是一种轻量级的数据传输格式，可以在多种语言之间进行数据交换，JSON 易于阅读和编码，而且是 JavaScript 规范的子集，能被支持 JavaScript 的浏览器所解析，虽然也是基于 XML 的，但是相比于 XML，JSON 编码简单，减少了解析时带来的性能和兼容性问题，这些特性使得 JSON 成为理想的数据交换语言，同时在 SDN 控制器和应用程序构建 API 方面，JSON 也被证明是一种优秀的语言[6]。

JSON 简单地说就是 JavaScript 中的对象和数组[7]，所以 JSON 具有对象和数组两种结构，通过这两种结构可以表示各种复杂的结构。对象在 JSON 中表示为"{}"括起来的内容，数据结构为{key：value，key：value，…}的键值对的结构，在面向对象的语言中，key 为对象的属性，value 为对应的属性值，所以很容易理解；数组在 JSON 中是中括号括起来的内容，数据结构为["java"，"javascript"，"vb"，…]，取值方式和所有语言中一样，使用索引获取。因此，这使得数据格式在同样基于这些结构的编程语言之间的交换成为可能。

从安全性上来讲，JSON 本来是 JavaScript 的一个安全的子集，不会含有赋值和调用，因此在将 JSON 数据装换成 JavaScript 对象的时候，包括许多 JavaScript 库都使用 Eval 函数，这意味着获取的 JSON 数据将被解析并执行，注意是执行，尤其有一些数据是来自用户输入的话，可能会带来意想不到的安全性问题。攻击者也可以利用这一点发送恶意的 JSON 数据，这样 Eval 函数就会执行这些恶意代码。

4.1.4 控制与转发分离——软件定义网络

根据上一节分析可以看到，网络工程师通常会使用命令行接口(CLI)或图形化用户界面(GUI)来配置网络设备。虽然这种工作方式很普遍，但是这里也有一些问题。实现复杂的网络配置可能要求工程师分别配置几个不同的网络设备。这个过程非常耗费时间、繁杂且容易出错。系统管理员可以使用一些自动化工具(如 Puppet)来简化他们的工作。尽管简单网络管理协议(SNMP)无处不在，但是网络工程师从未有一些能够处理任务分配的可预言管理工具。网络可编程性旨在转变原先的做法：通过提供应用编程接口(API)，通过编程语言向网络设备发送强大的编程指令。使用这种 API，就意味着网络分配不需要网络工程师通过发送 CLI 来实现，而是通过一些支持网络编程的工具来实现。例如，网络工程师可以使用脚本创建自动化分配任务，或者收集网络统计信息(虽然网络工程师现在仍然可以使用脚本，但是他们通常只是对 CLI 或 SNMP 交互的一种封

装而已),API 可以提供更丰富的功能,也可以为工程师创造可以协作的生态系统。软件定义网络的出现伴随着控制与转发分离、开放性可编程接口与集中控制,为网络管理提供了新的模式。

"软件定义网络"的主要见解是允许横向集成系统实现控制平面和数据平面的分离,同时提供日益复杂的抽象集。SDN 已经通过网络可编程性彻底改变了互联网行业。SDN 将改善和简化网络管理和操作。学术领域已经证明,SDN 在很多工程项目试验中取得了巨大的成功。最近也发现了 SDN 在工业部署方面比较成功。尽管 SDN 和活跃的网络模式的目的相同,即创建可编程的网络,但是它们两个的侧重点是不同的:活跃网络重点在于数据平面编程,而 SDN 的重点是控制平面编程。

尽管在 2009 年才创造了 SDN 这个词,但是 SDN 这个想法由来已久。它是在通用的可编程网络领域的许多不同的想法和建议的集合,最初 OpenSig 和活跃网络提出的可编程网络形成现在成熟的 SDN 架构。虽然 SDN 架构非常类似于主动网络架构,但技术上的优越性使它更受欢迎,而且它有更引人注目的实用案例,更重要的是,它有某些务实的设计选择。而且 SDN 变得流行很大程度上是由于在现代数据中心虚拟化和云计算的需要,由于它们依赖自动化配置和自动化编排,所以需要网络虚拟化支持。然而主动网络致力于开发全新的数据平面抽象,SDN 方法更侧重于新的控制平面抽象(可以说是解决一个更大的难点)。SDN 架构不同于主动网络架构,前者强调控制平面和数据平面的分离,而这并不属于主动网络架构的部分。

SDN 的日益流行,造成了各种工业利益相关者在 SDN 潮流开发初期的成功,并且 SDN 有着极其广阔的前景。通过分析和推断 SDN,需要对其进行精确的定义。SDN 有三个关键特征,首先是控制与转发分离;其次为集中控制,即一个单一的控制平面(或控制器)可以控制多个数据层(或交换机/路由器的数据);最后,SDN 结合模块化的控制平面高层抽象可以支持网络编程控制,即开放性可编程接口。传统网络与 SDN 网络架构的区别为 SDN 提出了网络抽象和创建通用的硬件设备,采取横向网络平台的机制。

通过提供抽象层,可以在控制器上编写新的应用程序实现更多的功能。这些应用程序包括建立虚拟网络、加强服务质量(QoS)以及显式路由等。SDN 应用最多的领域就是数据中心和校园网络。然而,SDN 也已经在许多其他网络中被提出,如服务提供商、运营商网络、无线网络。具体来说,SDN 已经被应用到无线传感器网络和无线网状网络的各项设置中,成为一种新的网络管理方式。

虽然广义而言网络一直是软件定义,但配置、管理和更新网络软件只能通过受雇于供应商的开发人员。这种"封闭"网络体系结构抑制了创新网络固有的不可编程和所有新特性请求被路由送到网络供应商实现。SDN 通过简单的但功能强大的开放网络改变这种模式,分离控制平面和数据平面。这种分离,再加上新控制抽象,形成新的 SDN 架构的核心。SDN 发展受到开源社区的支持。随着控制平面与数据平面的分离,第三方/开源开发人员为控制器编写程序成为了可能。这允许网络采用可编程硬件而不是封闭的硬件供应商,增加灵活性,同时降低成本。大体来讲,在 SDN 架构中有两个 API 接口,一个是北向接口,一个是南向接口。南向 API 结构定义了一个集中式网络控制器和网络设备之间的接口,北向接口 API 定义接口公开的网络应用程序控制器。互联网工程任务组(IETF)一直致力于标准化 SDN(I2RS 和 ALTO 项目)。此外,还有 Linux 基金会下的

OpenDaylight 项目，这是一个社区和免费框架，旨在培养创新 SDN 空间通过开源开发的最佳实践。

OpenFlow 是与 SDN 有关的典型协议。标准化 OpenFlow 协议作为 SDN 控制层抽象的准则并且推动了众多的创新。随着控制逻辑在一个单独的控制器实现与控制平面和数据平面之间的 API 接口标准化，使用高级编程网络可以实现网络编程可视化。通过 OpenFlow 协议定义的 API 接口是很原始的（与已有的汇编语言相比），更高层次控制语言开发与提供更高级的抽象只是一个时间问题。实际上，在这个方向的工作已经在开展中。看似无害的个人设备重构功能集中控制器释放了一个强大的新范式，提供有利于简化的抽象、高效和可扩展的网络运营管理和服务。

SDN 架构的发展也促进了网络操作系统抽象的发展。操作系统（OS）的角色是管理各种组件的复杂性、计算机组成、并给程序员提供一个简化的编程接口。NOS 通常实现于 SDN 的控制层。NOS 可用于管理层的实现，用于管理网络中的分布状态以提供持续的网络状态反映，并提供网络应用 API 作为更高一级开发的接口。各种 NOS 已实施 SDN 包括 NOX、ONIX 和 ONOS 后续工作的开创性工作。由于 SDN 架构在 2008 年的提案，许多研究工作都集中在中高层协议和应用程序以及对 SDN 架构所提供的可编程能力的发展。特别是，路由、传输层和管理框架已经提出用 OpenFlow 和 SDN 工作。路由建议包括 Quagflow，它的合作伙伴开源路由软件 Quaqqa 可以用它来提供在 SDN 环境下“虚拟路由器作为一种服务”。传输层协议建议包括打开 TCP 的工作。最后，出现了在支持与所述的 QoS 的多媒体传送工作。

在 SDN 架构中，如前面所述，使用了许多早期项目的可编程概念。尤其是，它是建立在早期建议的基础上：分离控制平面和数据平面（早期的建议包括 SoftRouter、4D、RCP 和 ForCES 工作组）；由一个单独的控制器（如 Tempest 框架）控制多个数据面；以及利用控制器和数据平面的开放接口（如 OpenSig 框架之间的通信）。

尽管 SDN 事实上利用了很多主动网络的工程，SDN 已经明显比它的前辈更受欢迎，由于各种技术推动（例如，计算机和网络技术的进步）和应用（例如，数据中心和云服务、网络虚拟化等），以及较大程度的工业接受，由于某些实用设计选择，SDN 将需要其控制平面和数据平面的概念创新。

4.2 软件定义无线网络

可编程网络的重要特性“软件定义”在无线网络中具有重要的应用前景，可以预见，无线网络将明确地向“软件定义”发展，结合软件定义特性的无线网络具有巨大的潜力。本节首先对已有的无线网络的可编程技术和结构——软件定义无线电与认知无线电进行简单介绍，然后讨论结合了 SDN 的无线网络的发展趋势。

4.2.1 软件定义无线电与认知无线电

1. 软件定义无线电

软件定义无线电（software-defined radios，SDR），简称软件无线电，是一种使用一个

简单的终端设备通过软件重配置来支持各种不同种类的无线系统和服务(包括2G、3G移动系统和无线LAN)的新技术。如图4-9所示为软件定义无线电与传统无线电的结构对比,主要体现在软硬件分割方面的不同。比如说,一台手提的或移动的SDR设备能让使用者通过改变软件而不是硬件改变发送和接收特性,如调制形式、宽带或窄带操作、发射功率和空气接口等。传统的陆上移动无线电设备是基于晶体管和集成电路的特殊硬件设备。更近代的陆上移动无线电设备使用数字信号处理器(DSP)芯片来处理无线电传输上的模/数和数/模转换。DSP芯片是一种实时数字信号处理器,能够通过执行不同的软件算法改变功能。当前先进的DSP芯片设计和制造技术使DSP芯片支持更多的功能。这些先进的DSP芯片使SDR设备具备更多的功能。SDR设备能够通过改变运行模式与存在的和演化中的LMR系统互通互联,并能够在多种公共安全频带中工作。软件定义无线电除了进行基本的无线通信功能外,还具备以下功能:①支持不同电台系统互联互通,使原来独立运行的不同电台系统能传递信息;②将通信业务功能从硬件方式解放出来,仅通过装载不同软件来动态配置系统功能,实现系统硬件平台的通用化、模块化和软件的装备化、模块化;③通过升级系统所装载的软件来达到对系统功能的更新和升级。

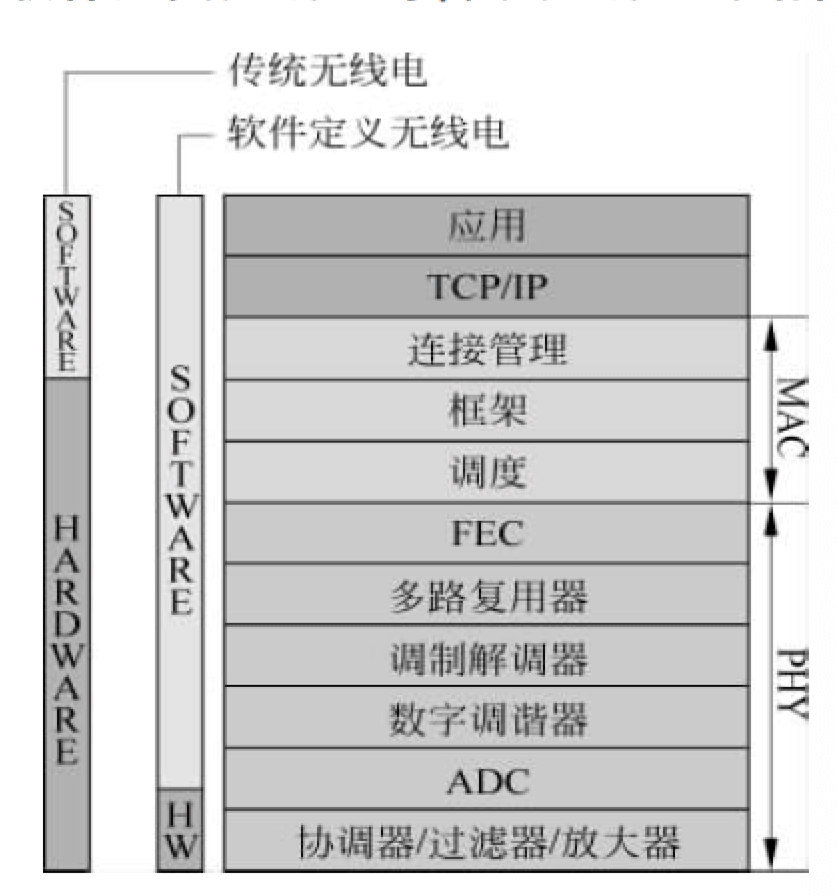


图4-9 软件无线电与传统无线电结构对比

总之,SDR能够让用户通过动态修改配置在系统中的软件更新设备和网络装备,从而获得更好的性能和服务。SDR设想为未来的可编程无线设备的一个基本组成部分,其重要性主要体现在,几乎所有的先进无线可编程技术(例如,认知无线电和可编程无线处理器等)都是基于软件定义无线电的。

2. 认知无线电

认知无线电(cognitive radio,CR)是指一种包含一个智能收发器的无线通信技术,该收发器能智能检测出哪些波段未被占用以及哪些波段正在被使用,当检测出某些波段未被占用时,CR系统就可以暂时使用该波段进行通信。认知无线电的概念最早由瑞典Joseph Mitola博士于1999年提出,是对软件无线电功能的进一步扩展。认知无线电可以感知周围电磁环境,通过无线电知识描述语言与通信网络进行智能交流,并实时调整参数(通信频率、发送功率、调制方式、编码体制等),使通信系统的无线电参数不仅与规则相适应,而且能与环境相匹配,以达到无论何时何地都能保持通信系统的高可靠性和频谱利用的高效性。作为一种更智能的频谱共享技术,CR是具有频谱感知能力的智能化软件无线电,理论上允许在时间、频率以及空间上进行多维的频谱复用,从而大大降低频谱和带宽限制对无线技术发展的束缚,因此,这一技术被预言为未来最热门的无线技术。自从认知无线电提出后,国内外许多学者和机构对其进行了全面、深入的研究。

认知无线电具备认知能力和重构能力两个基本特征。

1）认知能力

认知能力使认知无线电能够从其工作的无线环境中捕获或者感知信息，从而可以标识特定时间和空间内未使用的频谱资源（频谱空穴），并选择最适当的频谱和工作参数。这一任务主要包括频谱感知、频谱分析和频谱判定三个步骤。频谱感知的主要功能是监测可用频段、检测频谱空穴；频谱分析估计频谱感知获取的频谱空穴特性；频谱判定根据频谱空穴的特性和用户需求选择合适的频段传输数据。

2）重构能力

重构能力使得认知无线电设备可以根据无线环境动态编程，从而允许认知无线电设备采用不同的无线传输技术收发数据。在不对频谱授权用户产生有害干扰的前提下，利用授权系统的空闲频谱提供可靠的通信服务，这是重构的核心思想。当该频段被授权用户使用时，认知无线电有两种应对方式：一是切换到其他空闲频段进行通信；二是继续使用该频段，但改变发射功率或者调制方案，以避免对授权用户造成有害干扰。

尽管可编程的无线设备（如 SDR 和 CR）已经作为构建可编程的无线网络基础设施，应该提到的是，建立可编程的无线网络工作的任务需要更加细致以解决各种棘手的问题（如路由、安全性等），同时需要考虑广泛上的网络行为。历史上，大多数的认知无线电的研究都集中在器件级别的优化上，与网络级别的可编程特性相结合是最近才出现的。下面将介绍软件定义网络工程的发展趋势，并且认知无线电如何允许我们扩展可编程的概念。

4.2.2　无线网络的软件定义趋势

在本节中，主要探讨无线网络发展中一些显著的趋势，它们在创造未来可编程无线网络中具有发挥主要作用的潜力。这里将重点讨论软件定义无线网络（software-defined wireless networks，SWNs）、认知无线网络（cognitive wireless networks，CWNs），以及基于云的无线网络（cloud-based wireless networks，CbWNs）。通常的，对无线网络的一般处理都会归于对从电信演进而来的技术（例如以 WiMAX 和 LTE 技术为代表的 4G 网络）和具有主要数据网络基础的技术（如 Wi-Fi）这两种技术的讨论。

1．趋势一：软件定义无线网络

随着越来越多的无线网络设备部署和无线技术的多样化，管理无线网络已经变得非常具有挑战性。SDN 是一种很有前途的体系架构，可以用于方便地操作、控制、管理无线网络。如之前提到的，SDN 的典型特征普遍被认为是控制层和数据层的分离。可编程的控制器使得这种网络被称为是“软件定义的”。

可编程无线网络的发展，同时需要控制层和数据层的改进。尤其在无线环境中，数据层需要被重新设计，以定义出新的、更有效的抽象结果。让我们更清楚的解释它：当前支持 OpenFlow 的交换机对数据层的抽象是建立在原始的“匹配行为”的范式上的，为了给予控制层功能以更好的支持，数据层功能需要进一步发展，以支持更为复杂而实用的抽象。随着可编程硬件的大力发展和应用，对新一代的数据层抽象的研究正在如火如荼地进行着。因此，可编程无线网愿景的实现不仅需要多种相关网域的协作，更需要无线

网络数据层与控制层的协同创新。在无线网络环境中应用可编程网络技术可以将 SDN 的好处(简捷、灵活、进化能力及快速创新性)拓展到无线场景中。

下面首先讨论软件定义无线网络的应用。然后将讨论满足软件可编程的无线数据层的技术架构,以及在不同无线网络设置下 SDN 技术的应用。而更精细的细节是依赖于应用的,所有软件定义无线网络技术都在致力于实现:提供更简单的网络管理;允许相同硬件支持多种无线协议;提供一个抽象层使得全部或部分无线架构可以实现可编程性。正是控制和数据层的分离,使得单独的控制器可以动态重配网络属性,让上述目标更易实现。下面所讨论的软件定义无线网络方法中,渗透着为可编程性提供抽象处理这一主题。

SDN 最大的前景就是简化和提升网络管理及操作。关于这一点,接下来讨论 SDNs 的主要应用:

(1) 不同的技术共存。SDN 可以用来集成不同的无线技术以及促进多种无线技术的优化管理和共存。例如,Yap 等提出了利用 OpenFlow 无线项目中的 OpenFlow 协议来支持多种无线技术。SDN 可以通过支持移动分流,或 Wi-Fi 漫游[通过蜂窝移动网络流量外包给 Wi-Fi 数据网络来提高终端用户的经验质量(QoE)]来实现多种技术的共存。

(2) 移动分流(mobile offloading)。SDN 还可促进异构技术之间以及服务提供商之间的切换。现已出现了在 3G 或 4G/LTE 等移动网络下使用基于 OpenFlow 的控制器管理移动分流应用的成果。目前有多种漫游方案,促进着均质 Wi-Fi 的切换(如 IEEE 802.11f、802.11k 及 802.11r)以及异构 Wi-Fi 的切换,诸如 IEEE 802.21。Yi 等提出了利用 SDN 技术来实现异构网络的切换。在另一个近期工作中,Ding 等提出了软分流(SoftOffload),作为基于 SDN 的实现移动流量分配的可编程框架。

(3) 高效的资源利用。我们已经指出,OpenFlow 无线可以被用来支持不同的无线技术。此外,在多种其他基于 SDN 的工作中,OpenFlow 无线可以促进更有效的资源利用和更好的基础设施共享。在之后的分析中将提到,无线网络虚拟化在资源利用率上的促进作用弥补了 SDN 的不足。

(4) 更好的 QoS/QoE。控制和数据平面的分离和集中式控制器的使用可促进网络范围内 QoS/QoE 策略的较好配置。作为一个特定实例,Sivaraman 等提出了利用 SDN 接口来虚拟化互联网服务提供商(ISPs)和家庭网络的接入基础设施,以通过用户信息流的动态控制共享来实现更好的 QoS/QoE。

(5) 流量工程和自适应路由。SDN 可以通过改善流量工程(谷歌流量工程 B4 工作中,流量工程能显著提高网络性能,正是一个极好的例子)来大大提高网络利用率。SDN 架构可以在无线和移动网络中促进高效的流量工程和/或自适应路由。

(6) 分布式移动性管理(DMM)。DMM 是一种架构框架,为了将移动 IP 服务从当前部署的移动核心网(由于其分层集控架构而具有严重的可伸缩性、可靠性问题)迁移到分布式操作的更具扩展性的模型中。现有的 IP 移动框架强制移动网络运营商部署中央实体(如归属代理、本地移动性锚、分组网关等)时,需对移动性管理的协调性负责。现行的架构受到次优路由、缺乏扩展性、缺乏可靠性等问题的困扰。基于 SDN 的解决方案可以用于实现部分分布式模型,其中控制层和数据层是分开管理的。使用 SDN 原理的控制/数据的分离使得移动采用最佳路由成为可能。

(7) 程序设计的网络服务调度。另一种不断增长的趋势是,如 Schulz-Zander 等在最近工作中所提出的,将 SDN 用于实现动态编程服务调度。

(8) 实时分析/重新配置。互联网界重新燃起了部署集中式 SDN 架构的兴趣,即基于 SDN 控制器(群)实时运算出的分析,对网络基础设施进行优化重配。Metsch 等在其最近工作中,提出了使用实时分析来对移动网络进行业务操作和管理。

(9) 安全性增强。在 SDN 架构中数据层和控制层的精确分离以及在中央控制器中控制功能的巩固使得企业级安全策略的实现成为可能。在该方面,Ding 等提出了基于 SDN 在无线移动网络中增强安全性的框架。为提高安全性的移动网络,SDN 技术的一些应用程序包括:“入侵检测系统”(IDS)的实现;防止拒绝服务(DOS)在无线边缘附近的攻击;实现移动网络的安全切换。

下面将分别讨论两种卓越的软件定义可编程无线数据层架构——OpenRadio 和 OpenRoads。

(1) OpenRadio。OpenRadio 系统定义了一种新颖的无线数据层设计,允许通过模块化和声明式编程接口,对整个无线栈进行编程。OpenRadio 提出将无线协议的功能重构成两个部分。处理层负责使用底层硬件,运用程序和算法来处理数据。另外,决策层负责对处理层处理后的数据进行逻辑决策。应当指出,处理层和决策层的概念分别巧妙地类似于 SDN 世界中的数据层和控制层。在 SDRs 和 SDNs 的模型中,OpenRadio 都是其中的主题之一。OpenRadio 使用用于管理通过类似 SDR 的软件配置通用硬件的无线协议的抽象层,同时也允许协议从类似 SDNs 硬件中分离。OpenRadio 可以支持不同的无线协议,如 Wi-Fi、WiMAX 和 LTE 等,从而能显著地降低成本并使其更容易配置、优化,甚至定义协议。OpenRadio 的主要优势在于它能从硬件上分离协议和用软件绑定之前的协议以增加灵活性。随着新的无线协议定期推出,集中重编程功能和可编程能力会提供极大的方便。OpenRadio 也可用于在蜂窝网络中基于小区尺寸的优化以及用于多个异构小区站中的频谱管理,以避免干扰。

(2) OpenRoads(或“无线 OpenFlow”)。可编程 SWNs 领域的一个开创性发展就是 OpenRoads 项目,也被称为 OpenFlow Wireless(OpenFlow 无线)。OpenRoads 提供了可用于在无线环境中应用 SDN 原则,从而创造一个可编程无线数据层的完整平台。OpenRoads 的一个特别有吸引力的好处是,它允许不同的无线技术之间的有效切换,通过“平坦化”多元垂直整合的无线技术,支持移动无线网客户的无缝移动。Yap 等对这种基于 SDN 的方法的可行性进行了探讨,研究了 IEEE 802.11 和 IEEE 802.16 网络之间垂直切换的移动性管理。OpenRoads 在无线路由器上采用了 OpenFlow 协议和简单网络管理协议(SNMP)。OpenFlow 提供了管理转发层的方法,而 SNMP 允许了这些无线设备的配置。FlowVisor 和 SNMP 解复用器是用来分块的,从而使控制具有更强的可扩展性。每个控制流量被赋予一个特定的“流量值”,以确保不同的控制流互相独立,使得只有对应特定设备的控制流才会被加载。被应用在 OpenFlow 上的高级别的控制接口提供了不同的设备之间的通信、无线设备的配置,以及流量的管理。

现在将详述包含了 SDN 原则的不同无线网络项目。这些项目在如何应用 SDN 原则的方法及无线网络[无线本地局域网(WLAN)、蜂窝网络、传感器网络及个人局域网]的性质上各不相同。

(1) 基于 WLAN 的 SWN。Odin 是一个已提出的将 SDNs 原则运用于 WLANs 的 SWN 架构。一个受欢迎的表达是，WLAN 决策不是由 WLAN 基础设施本身做出而是由用户设备做出的。举例来说，用户设备可以根据自己的偏好来决定要接入哪一个接入点，而非由设施来为用户决定。在无线局域网中，客户端和特定接入点的关联对客户移动性构成了挑战。这对潜在的面向 SDN 的 WLAN 基础设施造成了显著的挑战，因为这会对控制器程序员如何追踪接入点和客户端之间千变万化的关联造成困难。Odin 架构提出了光虚拟接入点（LVAP）的使用。LVAPs 把接入点—客户端的关联虚拟化，从而从物理接入点中把关联解耦。每当一个客户端连接到 WLAN 网络，它就被分配一个与物理所接点无关的固定 LVAP 识别号。因此物理接入点的复杂性对中央控制器程序员不可见。Odin 程序有很多优点。在物理接入点变化时新连接的频繁新建条件下，它提供了接入点之间的无缝移动性。此外，灵活的路由策略得以进一步允许负载均衡。而且，随着网络概观的改进，减少干扰并排除问题（诸如隐藏节点问题）成为可能。

(2) 移动蜂窝软件定义网络。最近，人们对利用 SDN 原理来提高移动蜂窝网络的性能表现出了显著的兴趣。特别是，研究者已经提出了将 SDN 原理掺入 3GPP 演进分组核心的移动运营商网络（MobileFlow 项目）和 4G LTE 蜂窝网络（SoftRAN 项目和 SoftCell 项目）的框架。移动蜂窝 SWN 方法的主要优点包括对无线资源更好的管理、更灵活的路由、实时监控、更好的移动性支持以及向 Wi-Fi 网络分流的能力。当前的 LTE 架构有一些问题：集中式数据流，因为所有数据都从分组网关（P-GW）中通过；集中的监控和控制都难以扩展而且昂贵；基站和基础设施难以配置。前两个问题都与 LTE 网络的中央控制和监控有关。一个可行的解决办法需要将控制和监控的责任分发下去，构建一个混合控制平面。这看上去是对 SDN 基本原则的背离，如集中控制。对第三个问题的解决方法需要将基于 SDN 的架构调整到能实现远程应用程序解决任务。如前所述，OpenRadio 提供了理想的模块化接口来远程且方便地配置基站。

(3) 基于无线传感器网络（wireless sensor network，WSN）的软件定义网络。无线传感器网络一直流行于研究界，但也一直被认为是一个专用于应用程序的技术。WSN 作为专用应用程序这一点，导致了在同一区域部署多个 WSN 时资源利用率低下的问题将 SDN 与无线传感器网络结合可以为上述问题提供解决方案。控制层和数据层的分离会提供管理和控制网络所需的抽象化。通过部署基于传感器网络的软件定义网络，网络控制者可以设置策略和服务质量来支持多种潜在的应用。这也可以允许一个传感器节点服务于多个应用或需求。这又能进一步提高资源利用率和实现最优化。

(4) 基于低速率个人区域网（low-rate personal area network，LR-PAN）的软件定义网络。SDN 的特性也可以在低速率个人区域网中发挥很大优势。所有的 LR-PAN 基本上使用的都是相同的 802.15 数据链路层。在它们各自协议栈中较高层的差异导致了 LR-PAN 协议的不同，如 ZigBee、Bluetooth 等。这使得不同节点之间的通信不兼容。通过使用和 OpenRadio 所使用的相同的工具，可以将协议中的硬件部分分离出来，使用一个抽象层去编写和定义不同的无线协议。这使我们可以在相同的无线设备上运行不同无线 LR-PAN 协议。这样，节点就可以同时动态与多种网络相连，使我们可以更有效地利用网络资源。数据层和控制层的分离使得 SDN 简单管理、灵活控制、有效资源利用的优势扩展到低速率个人区域网中。

2. 趋势二：认知无线网络

前面已经强调指出，大多数现有认知无线电研究的主要重点在于实现智能设备级的行为，而在认知网络上的一些工作明显是例外的。与认知无线电相比，认知网络具有拥有网络级别的智能及自感知行为的特性。下面将这种认知网络称为“认知无线网络”(CWN)。CWN 采用了认知循环(如图 4-10 所示)，以观察环境，定位自身，以及之后根据网络/用户和应用的上下文来做出决定/计划，以达成最优决策。

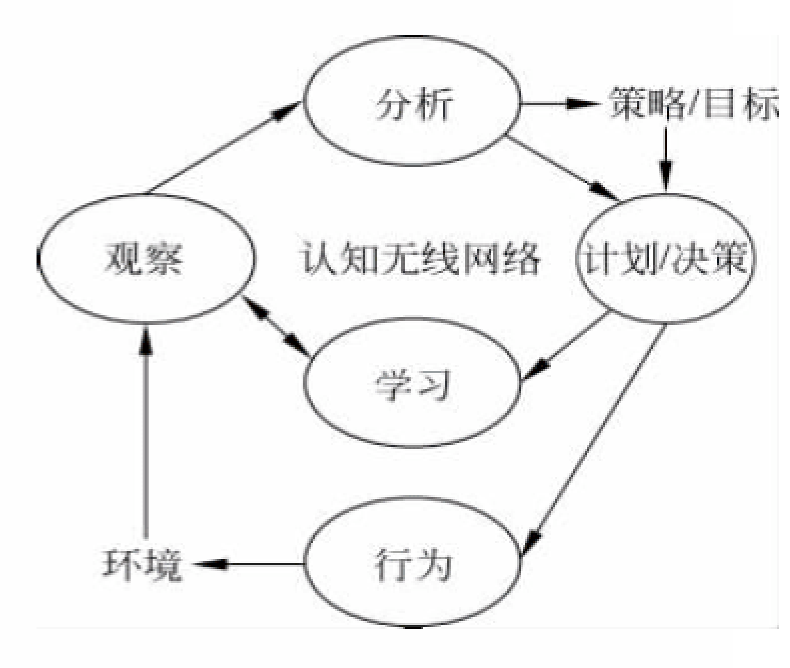


图 4-10　认知无线网络

虽然设备级重新配置能力(如 SDR 和 CR)和网络级重新配置功能(如 SDN)无疑将成为未来可编程无线网络的重要组成部分，由此产生的可编程无线架构仍然不会是完全自动化的，除非将人工智能技术掺入到框架的核心。除了提供了各种有用抽象的可编程数据和控制这两个平面之外，未来的无线可编程网络还需要一个知识平面。由于 CRNs 本身在无线通信方面就包含了人工智能技术，它很自然地可以与 CRs 以及 SDN 和 SDRs 的功能一起探索，并为未来可编程无线设备提供实现知识平面的构建机制。

在今后的工作中，混合使用 SDN 和 CRN 技术更有可能衍生出更强大的可编程无线网络的范例。虽然 CogNet 项目曾提出过一个分离控制和数据平面的架构模型与一个可扩展的全球控制平面通过一个 API 控制分离的数据平面(这和 SDN 架构的精神相似)，只是这个初始概念没有后续的跟进。这个领域中，对如何发挥 CRN 和 SDN 的优势，探索条件已经成熟。

在这里注意到 CWN 是自主的自编程网络，例如，CWN 吸收了自动自我调整或编程的能力，可以最优化操作参数，以满足所需的性能目标。CWN 中可编程性的概念与传统观念中隐含假定了非自主性的可编程性(这也适用于 SWN)的概念不同，未来可编程无线网络可以同时采用自主和非自主的编程以获得两种方法的优点。

下面介绍 CWNs 的应用和 CWNs 的一个特定网络分类。

1) 认知无线网络的应用

(1) 动态频谱接入。CWNs 的一个重要自适应特征是动态频谱接入(DSA)，它允许 SU 的工作频率重新配置，允许在许可的频谱进行通信。这主要取决于频谱感知(为检测主用户，即 PU，而执行)，这是用于确保以 PU 为代表的在许可期内的用户不受干扰。在某些情况下，频谱感知可以被避免，只需要数据库查找指定 PU 的活动模式就可以了。许多 IEEE 标准(如 IEEE 802.11、802.15 和 802.16)引入了一些基础的感知无线电功能，如动态频率选择(DFS)、功率控制(PC)，以促进共享频段的网络的共存。而 DSA 是 CRNs 最普遍引用的应用，开发 CRNs 的网络级的智能会促进许多其他的应用，包括根据网络条件最优地重新对自身编程的能力。

(2) 认知网络。认知网络广泛地包括了认知和学习的模型，在强调端对端全网范围时为 CRSs 进行过定义。这种认知网络可以感知当前条件来计划、决定并迎合整体网络

端至端的目标行动。图 4-10 表明了认知网络的愿景。要想将 CRN 改造为认知网络(CN),需要将智能整合到 CRN 架构和跨栈协议的构造中去。认知网络的愿景是一个智能网络,给定高优先级的命令可以自启动,并且为了最优化网络整体性能度量值而能够不断调整和管理自己。认知网络的宏伟愿景是可以搭载很多有趣的应用。特别的,网络优化的 MAC 协议、自适应路由以及改进的安全性都有特别的前景,可以在无线网络服务领域前沿做出重大的进展。

(3) 参数优化。在典型的无线网络中,已经观察到物理和 MAC 层提供了许多"可以拧的旋钮",以优化其性能,可测范围为几米。在以往的研究中,以 CWNs 为环境,提供了许多关于旋钮与几米的物理层和 MAC 层的例子。由于无线网络经常在动态条件下运作,优化配置旋钮不是一个简单的问题。各种基于 AI 的技术已在文献中被提出,以协助 CWNs 满足在这样的设置下能执行自主参数优化调整的需求。除了人工智能,CWN 还从各领域借鉴技术和工具,如博弈论、控制论、优化理论、元启发式算法等。

(4) 增强的可靠性。为了被广泛部署,下一代无线网络要独立于网络状态和无线电传送介质,提供增强的可靠性。不幸的是,无线传送介质尤为容易出错。CR 被认为可以解决这个问题,它们可以通过有效的错误恢复和预防机制来对抗各种出错。这可以使 CWNs 在各种环境下都可以提供一致的服务质量。相关详细调查[8]阐述了各种故障情况及 CR 如何应对这些故障而增强可靠性的方法。

(5) 自适应优化。CWNs 提供了利用基于 AI 技术的应对各种无线环境条件的自适应优化能力。CWN 通常利用自适应框架,如强化学习、学习自动机、博弈论等,用来自适应优化网络参数。针对 CWNs,有大量的基于优化的应用,包括动态频谱接入、参数优化、优化 MAC 和路由、增强可靠性和安全、QoS 保证和管理、信道分配等。

2) 认知无线网络的特定网络分类

认知无线电技术的使用已经在各种网络设定中被提出。这使得 CWN 具有一种趋势,可以在不同网络设定中将自己列上名单,并且每次都提供认知网络的优点。目前已经有将认知技术应用到 IEEE 802.11 网络的工作。Niyato 等提出了在基于 IEEE 802.11 协议的无线 Mesh 网络中使用 CR 来进行机会信道选择。此外,还有用商用的 IEEE 802.11 WLAN 网卡来实现"空白频谱"网络的工作。CR 也被用在为了用于电视频段中而提出的 IEEE 802.22 无线局域网标准中。其他"空白频谱"网络项目还包括 Yuan 等的工作,专注于让 DSA 使用电视频段中的空白频谱。还有将 CR 技术嵌入无线传感器网络(WSN)和车载网络(VANETs)的工作,以分别帮助实现认知传感器网络和认知车载网络。

4.2.3 无线网络虚拟化

虚拟化技术通过对原始资源一致抽象化实现了多种不同的计算机技术的资源共享,并在此基础上实现了多种实际功能。虚拟化技术已经在当前大规模数据中心时代产生了不可估量的影响,在虚拟化技术得到广泛应用之前,出于对安全、隔离、性能等方面的考虑,指定出用于特殊应用的服务器(如网络服务器、数据库服务器等),同时为峰值负载进行预分配,这导致资源未得到充分利用成为常见现象。这一原因促进了"虚拟机"

(virtual machine)这一抽象概念的诞生，多种虚拟机实例中，虚拟机创建于同一物理机器上，每个虚拟机互相完全隔离。这些虚拟机将底层物理服务器统一化，为端应用提供了接口。虚拟机具有可复制可移动的可编程特性，允许将虚拟机迁移到任意物理服务器上，这一特性使得我们可以高效安全地共享物理资源。因此，虚拟化已经成为现代计算机技术发展过程中不可或缺的一部分。

如今虚拟化应用已经从操作系统方面的节能与计算产业经济转移到数据中心环境，随着虚拟化网络扮演着越来越重要的角色，未来可编程无线网络的发展将以虚拟化为基础。这一趋势同样引起了网络供应商的兴趣，这意味着将电信设备的功能从专用设备中解耦出来，实现了网络功能虚拟化(network functions virtualization，NFV)。如图4-11所示是一个网络功能虚拟化将网络功能从固定功能的硬件电气设备上转移到了虚拟云软件上的实例。在固定或者移动网络的应用场景下，扩展到任何数据平面的数据包处理功能和控制平面功能中的NFV概念，包括但不局限于路由、交换机、家庭网关等这一类的移动网络节点和传统的交换设备。同时，NFV技术允许供应商利用数据平面可编程，因此更容易高效率地实现可以统筹管理的中间设备的功能。

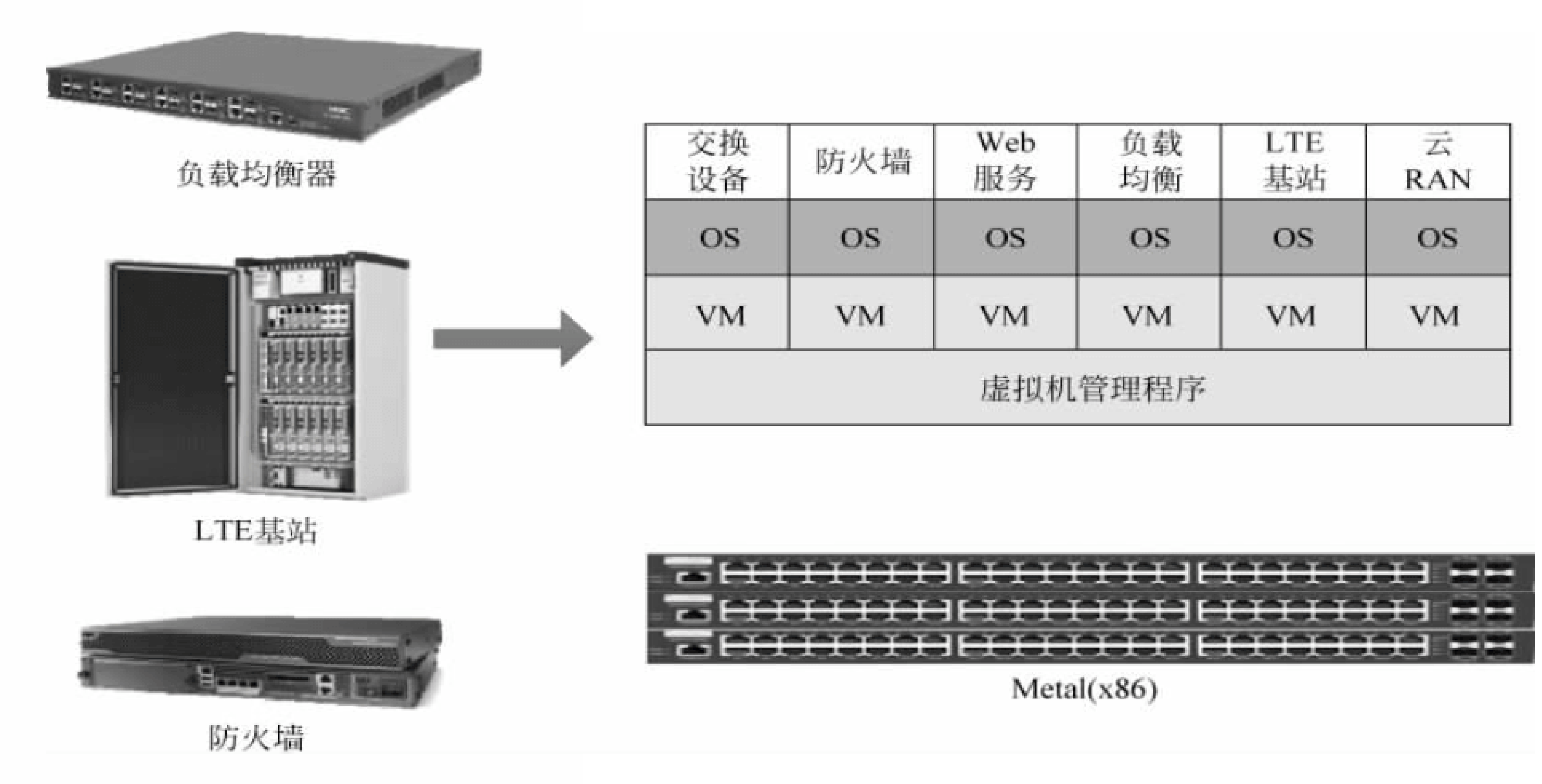


图4-11　NFV在无线网络中的应用

在传统网络中，单独数据包在数据平面通过多个工作于第二层和第三层的交换功能的中间设备被处理的情况并不少见。但是，传统中间设备大多基于安置在网络中位置固定的集成电路平台上。这种传统的架构具有僵硬、静态等缺点，并且无法适应自动化和可编配的趋势。与之相对的，网络基础架构的可编配功能可以在NFV的架构上实现。另外，通过动态的“网络服务链”(network service chaining，NSC)，虚拟网络功能(如ACL、QoS、负载均衡等)作为虚拟化的云实例运行在数据中心。

在利用SDN概念的基础上实现网络虚拟化解决方案方面，相关研发者已经完成了大量的工作。一个最初的SDN用例允许研究人员在工作网络的虚拟“切片”上运行实验协议。同时通过SDN通过切片网络实现网络虚拟化的概念已经应用于VINI、PlanetLab、Emulab以及最近在NSF资助下的全球环境网络创新项目(global environment for network

innovation,GENI)Casado 等。目前已经提出的一个网络管理程序的虚拟化网络的转发平面进一步扩展了这个想法。一个网络管理程序的概念类似于传统的虚拟机管理程序的概念,网络管理程序实现一个全网范围内的软件层,其旨在支持多个虚拟网络。换句话说,即将网络从底层硬件设备中解耦。并且因此网络可以像虚拟机一样被创建、移植和删除,这一切都仅仅依靠软件就能实现。基于 OpenFlow 的 SDN 环境下的网络管理程序 FlowVisor 允许彼此之间相互隔离的虚拟网络切片,并且每个切片由一个单独的控制器控制。

除了 NFV 之外,利用 SDN 打造服务链还有一些工作,通过一个可重复使用的平台 OpenPipe,OpenFlow 被用于设计和原型化一个高速网络。使用一个 OpenFlow 网络,新系统可以被快速构建,就像 Click 模型化的路由一样,通过无论在 CPU、FPGA 还是 ASIC 上实现的"管道"模块,整合为一个管道。OpenPipe 还可以灵活地将模块从一个子系统移植到另一个上面(通过软件或者硬件或者两者同时作用来实现)。

总之,最初从数据中心中提出的云计算概念很可能在未来创造可编程无线网络中发挥重要作用。预计最近几年中,移动运营商和电信网络将越来越多地效仿他们在数据中心和云方面的虚拟化技术、开放性软件接口和依赖数据中心和目前方案中的商用服务,实现垂直一体化硬件设备的部署。特别的,云计算和网络虚拟化的结合允许网络可编程性的实现导致了前所未有的灵活性,主要体现在可以快捷创建、部署和虚拟化设备,并且按照用户的需求进行管理的新服务。这可以创建一个新的面向服务的无线网络架构,并允许异构无线网接入技术结合云基础架构。

4.3 软件定义数据中心

4.3.1 软件定义数据中心简介

软件定义网络的另一个比较成功的应用场景是数据中心。近年来,任何一个成功的云服务商或者互联网服务商的数据中心的创新与技术均依靠软件来完成,数据中心的显著特点是硬件设备比较统一,于是通过软件进行有效的分配和调度资源成为需要解决的关键问题,而数据中心中,物理资源主要包括计算、存储和网络资源。本节首先对软件定义数据中心进行概述,然后介绍软件定义存储与软件定义计算,最后探讨软件定义数据中心现状。

1. 数据中心的发展历史

数据中心又称为服务器农场(server farm),它用于容纳计算机系统和相关的组件,如通信和存储系统,通常包括冗余的或备用的电力设备、冗余的数据通信连接以及环境控制(如空调、消防设备)和安全设备,是企业信息化的重要基础设施。

数据中心的根源来自早期计算机领域巨大的计算机房。早期计算机系统体积非常大,本身就需要占用很大的空间,同时运行和维护也都很复杂,需要在一个特殊的环境中运行,因此需要许多电缆连接所有的组件,如标准机架安装设备、高架地板和电缆盘(或安装在屋顶或架空在地板下)。此外,过去的计算机也需要大量的电力,会产生大量的热

量，通过专用的计算机房和冷却系统可以对散热效果进行较好的控制。安全也很重要——那时计算机是很贵的，主要用于军事目的或重要的经济科研领域，因此对计算机的访问受到了严格的控制，比如在我国石油勘探行业就流传着“玻璃房子”的故事。美国为防止早期从其进口的计算机被用于军事用途，往往派美国人来管控计算机房，中国人上机还需要得到批准才行。因此，专用的计算机房是必须的，这就是数据中心早期机房的雏形。

到了20世纪80年代，微机市场一片繁荣，大量计算机被部署到世界各个角落，但很多时候很少有人关心过对这些计算机的运维要求。然而，随着IT运维变得越来越复杂，数据丢失现象越来越普遍，所有公司开始意识到需要控制IT资源。特别是随着20世纪90年代客户端/服务器计算模式的普及，微机（现在叫做“服务器”）开始在过去的机房中寻找它们的位置，使用廉价的网络设备，加上标准的网络布线，人们开始采用分层设计，将服务器单独放在公司的一间特殊的房间中，于是就使用“数据中心”一词表示特殊设计过的机房。大约就是这个时候“数据中心”这个词开始流行开来。

在互联网泡沫期间，数据中心得到了蓬勃发展，公司需要在互联网上建立可以不间断访问的网站，而这要求具有快速和可靠的互联网连接。于是出现了许多互联网数据中心（IDC）公司，这种IDC的建设是非常昂贵的——物理房屋、设备、训练有素的管理人员使得这些大型的数据中心成本非常高。IDC采用了大量的新技术和手段为运维提供了可伸缩的能力，这些做法最后成功迁移到企业的私有数据中心里。

截至2007年，数据中心的设计、建造和运维已经形成了非常规范的规程，如电信行业协会（TIA）等专业机构推出了被行业认可的设计标准。同时，数据中心的一些运行指标也被开发出来，用于评估中断对业务的影响。目前，仍然有大量的操作规程正在开发，而且绿色、节能、环保的数据中心也开始受到大家的广泛关注。

数据中心电信基础设施标准[9]描述了数据中心基础设施的要求，最简单的是一级数据中心，基本上就是一个根据基本准则安装计算机系统的机房。最严格的是四级数据中心，其设计目的是托管关键计算机系统，具有完全冗余的子系统和独立的安全区域，通过生物特征识别技术进行访问控制。另一个思路是将数据中心放在地下，这主要是出于数据安全和环境方面的考虑，如冷却要求。

数据中心的网络基础设施[10]：

（1）数据中心的通信目前通常是运行在基于IP的网络上，数据中心包含一套路由器和交换机在服务器和外界之间传输数据，为保证网络的稳定可靠，往往会提供两个或更多冗余的互联网连接。

（2）数据中心有些服务器用于运行基础的互联网服务和组织内部用户需要的内部网服务，如电子邮件服务器、代理服务器和DNS服务器。

（3）数据通常还需要部署网络安全设备，包括防火墙、VPN网关、入侵检测系统等，也包括监控网络和某些应用的监控系统，典型的场所监视系统也很普遍，在数据中心内部发生通信故障时可以快速判断故障位置。

2. 软件定义数据中心的解决方案

目前，IT基础设施及其运营越来越复杂，人们通常采用云计算和虚拟化技术来满足各种业务需求。在过去的10年里，服务器虚拟化重新部署、管理以及优化了计算资源，

将数据中心转化成为一个更加灵活高效的业务应用平台。专用服务器被动态托管之后，在虚拟服务器环境中能够根据需求运行应用程序。

虽然虚拟化重塑数据中心的运营，企业能够部署机架服务器汇集和分配应用程序的需求，但这种转变并不完整。数据中心网络和存储资产仍然保持着孤立和静态配置，很少有设施能够自动化统筹管理混合网络和存储硬件。

软件定义的数据中心（software defined data center，SDDC）声称改变这种状况。VMware对其描述为："一个统一的数据中心平台，提供了前所未有的自动化、灵活性和效率，并转变IT交付的方式。汇集和汇总计算、存储、网络、安全性等可用性服务，并交付软件，通过智能化的策略驱动的软件进行管理。"

软件定义数据中心的目标是使数据中心运营的许多方面受益：更有效地利用资源；更加容易配置和重新配置；以及更快地部署新的应用程序等。构建一个软件定义数据中心，主要包括两个核心要素：虚拟化数据中心的一切资源（资源虚拟化），通过一个核心的管理平台对这些资源进行管理（集中管理）。

软件定义数据中心首先要做的就是虚拟化一切物理资源，通过虚拟化技术构建一个由虚拟资源组成的虚拟资源池，包括服务器虚拟化、存储虚拟化和网络虚拟化，当前许多用户数据中心环境是异构的，同时包含了虚拟化设备与实体设备、私有云与公有云，因此支持异构云环境是软件定义数据中心的必要条件。而服务器在虚拟化应用的基础上，软件定义数据中心能够虚拟网络和存储资源，使抽象的数据中心的基础设施可以通过应用程序和软件进行访问。

SDN概念在软件定义数据中心之前提出，核心思想包括控制与转发分离。集中控制与开放性可编程接口，SDN为灵活控制流量、网络架构创新提供了良好的平台。在数据中心中，SDN实现了从虚拟网卡、虚拟交换机、接入交换机到核心交换机的控制与传输的分离。最终，软件定义数据中心将不再需要IT技术人员来操纵孤立的服务器，网络和存储硬件将响应请求。相反，配置自动进行定义的规则和框架、政策和服务水平协议（SLA），通过应用程序编程接口（API）调用的自动化和业务流程引擎，并从一个集中的环境内配置适当的资源。

综上所述，将SDN融入数据中心中，能够有效地提高数据中心的管理能力和运行效率。通过将数据中心网络纳入可控范围，管理层可以快速发现网络状态的变化并实时地进行调整，提高网络运行效率；通过将SDN与计算资源管理系统融合，使得整个数据中心设施有了统一的管理系统，提高了管理能力。管理层的操作都集中到控制器中，通过修改控制器的逻辑即可对数据中心进行配置，简化了管理操作。另外，软件化的实现使得管理人员可以灵活地对策略进行修改并实现多种多样的功能，如在网关上实现防火墙等，这既提高了管控能力也在一定程度上降低了成本。将SDN架构引入到数据中心已经成为一种具有广阔应用前景的软件定义数据中心解决方案。

4.3.2 软件定义存储

软件定义存储（software defined storage，SDS）是一种数据存储方式，所有存储相关的控制工作都仅在相对于物理存储硬件的外部软件中。这个软件不是作为存储设备中

的固件，而是在一个服务器上或者作为操作系统（OS）或 hypervisor 的一部分。

随着互联网技术的快速发展、云计算应用模式的快速普及以及大数据时代下数据的爆炸式增长，传统的存储系统越来越难以满足企业系统在灵活性、扩展性以及统一管理等方面的需求。数据中心面临数据爆发带来的挑战的同时，存储设备作为数据存储与交换的承载实体面临着诸多问题，急需进行技术革新。目前存储行业需要改善的主要问题有：要求存储资源具备良好的弹性扩展能力，消除单节点的存储能力瓶颈，保持无线的节点扩展性；有效整合各类异构存储资源，形成统一的资源池供应调用，进行异构存储资源的整合；要求数据中心的存储资源能够快速灵活地分配，自动和动态地根据需求进行调整，实现智能化的存储资源管理[11]。

为了打破传统存储系统软硬件紧耦合所造成的割裂状况，“软件定义存储”应运而生，这一概念最早由 VMware 在 2012 年提出，IDC 研究主任 Nadkarni 在一篇声明中提到，基于软件的平台会继续比基于文件或者基于对象的其他存储市场部分发展得更快，这种成长主要被跨行业和地域、丰富多样、密集的数据所驱动。目前国内研究机构对 SDS 的标准化工作还没提上日程，国际研究机构正在推动 SDS 的标准化，IDC 在 2013 年对外正式宣布发布文档对 SDS 的定义进行了阐述，并提出 SDS 的评估方法以及在通用硬件上利用软件构建一个高可扩展性的存储平台的解决方案[12]。OpenStorage 开源社区在 2013 年举办了 OpenStorage Summit 2013 大会，组织各个企业一起讨论 SDS 定义。在商业界，对 SDS 进行深入研究的公司主要包括传统的存储设备厂商（如 EMC 公司、华为技术和 NetApp 等）、虚拟化解决方案厂商（如 VMware）以及各个创业存储软件公司，由于其需求不同，研究侧重点也不尽相同。

从软件定义存储的需求和概念分析得到软件定义存储技术所应具备的特点：自服务，即通过定义标准的 API（应用编程接口），进行存储配置，满足应用所需的存储资源，无须人工干预；存储虚拟化，即通过存储虚拟化聚合异构的存储资源，形成一个共享的存储资源池，打破传统存储系统烟囱式的工作模式；软硬件解耦合，即软件定义存储允许运行在通用的硬件设备上，用户不必采用特定厂商的存储硬件；丰富的数据接口，即软件定义存储可以对外提供丰富的数据接口，如文件系统接口、块接口、对象接口以及可以供大数据分析。

从存储服务的整体角度来看，由底层的物理存储设备到上层的存储对外服务接口，软件定义存储的技术架构可以分为存储资源池、存储适配层、软件定义层、访问接口层 4 层。与网络虚拟化思路一致，首先对各种类型的存储资源进行抽象化和池化，形成不同类型的存储资源池；然后通过多种访问接口实现对底层存储资源池异构存储设备之间的访问，实现异构存储设备的整合；再次通过软件实现对底层存储资源的统一管理和调度；最后提供存储资源池对外的接口，主要包括管理接口和数据接口。管理接口主要提供与自服务门户、管理门户以及各类外部管理系统等的接口。数据接口通过协议转换的方式，为外部应用提供丰富的数据接口。

4.3.3 软件定义计算

软件定义的计算的目的是将相关的智能信息从硬件抽象到更规范的软件层。这不

仅能够使一个异构的数据中心更有效地工作，也使得 IT 团队能够更容易地引入新的功能。在云架构或高度虚拟化的数据中心，一款应用程序应该能够在一个广泛的 IT 设备集合中资源共享。例如，如果一家数据中心采用来自思科和 Juniper 网络公司的一种混合的网络交换机，思科的 IOS 和 Juniper 的 JunOS 工作主要的差异将存在于创建一个虚拟网络的难易程度方面。

计算机科学家 Andrew S. Tanenbaum 在其著作《Distributed Operating Systems》中提到的 5 点可以精确地描述目前软件定义计算的需求：透明性、灵活性、可靠性、性能要求以及伸缩性。这 5 点要求针对的是分布式系统，现实世界中，多台计算机组成的具有一定程度耦合的计算系统实际上都是分布式系统的一种，包含云计算、高性能计算在内，因此这 5 点要求包含了对实际上的计算系统的要求。

将 NFV、SDN 技术与数据中心的计算资源管理技术相结合，让计算机元件在虚拟的基础上而不是真实的基础上运行，可以扩大硬件的容量，简化软件的重新配置过程。CPU 的虚拟化技术可以单 CPU 模拟多 CPU 并行，允许一个平台同时运行多个操作系统，并且应用程序都可以在相互独立的空间内运行而互不影响，从而显著提高计算机的工作效率。

因此，软件定义计算将改变现在的传统 IT 架构，而且将互联网中的所有资源全部连在一起，形成一个大的计算中心，但用户却不用关心计算的具体进行过程，而只需关心提供给自己的服务是否正常。虽然软件定义计算技术具有巨大的应用前景，但是在可预见的未来这一过程还有很长的路要走，因为还没有哪种技术是不存在潜在缺陷甚至陷阱的。但是这一技术融合了云计算、下一代互联网、虚拟化、软件定义网络等领域的研究范畴，对于现有网络中的例如结构体系僵化、网络资源利用率低下、多用于网络资源分配不均匀等问题提供了开拓性的解决思路。

4.3.4 软件定义网络在数据中心中的应用分析

目前软件定义网络已经在数据中心实现了部署，而且相比于传统网络的管理控制机制，软件定义数据中心对于数据中心工作效率的提升非常大。谷歌 B4 网络架构[13]提出，在数据中心网络中，应用 SDN 使得链路利用率由 30%～40%提高到了接近 100%，证明了软件定义在提高资源利用率方面具有巨大的应用前景。这个网络一共分为三个层次，分别是物理设备层(switch hardware)、局部网络控制层(site controllers)和全局控制层(global)。一个 site 就是一个数据中心。第一层的物理交换机和第二层的 controller 在每个数据中心的内部出口的地方都有部署，而第三层的 SDN 网关和 TE 服务器则是在一个全局统一的控制地。Google 的部署分为三个阶段。第一阶段把 OpenFlow 交换机引入到网络里面，但这时 OpenFlow 交换机对同网络中的其他非 OpenFlow 设备表现得就像是传统交换机一样，只是网络协议都是在控制器上完成的，从外部行为来看表现得仍然像传统网络。第二阶段引入更多流量到 OpenFlow 网络中，并且开始引入 SDN 管理，让网络开始向 SDN 网络演变。第三个阶段整个 B4 网络完全切换到了 OpenFlow 网络，引入了流量工程，完全靠 OpenFlow 来规划流量路径，对网络流量进行极大的优化。

与 B4 系统基本原理类似，微软公司的 SWAN 系统[14]同样利用 SDN 体系结构实现

数据中心间高效的利用率。它实现的手段是：当通过SDN全网信息观测到某条链路需求较低时，SWAN控制数据层的数据通路迅速切换至该链路来传输数据，从而保持所有链路长时间的高效利用率。SDN技术保障了SWAN能够进行全局观测以及流量工程的合理运用，确保资源利用率长期处于60%以上。相对于B4系统，SWAN系统采用的是传统设备，便于设备的更新与维护，更利于该系统的普及。在数据中心网络中，可以利用SDN通过全局网络信息消除数据传输冗余，从而避免了网络资源的浪费。另外有研究者提出，将SDN应用在了数据中心路由中，利用SDN掌握全网信息，通过对流按时间进行调度，让每条流在某一时间段独占带宽，减小了流的竞争时间，使链路得到充分利用，从而节省了数据中心的能量。而利用了SDN掌握全局信息能力[15]，实时关闭暂未使用的设备，当有需要时再打开，节省了约一半的能耗。由此可见，利用SDN，可以在数据中心网络中实现网络资源的优化分配，还可以通过对业务流进行调度等方法，提高链路利用率，同时降低网络能耗，几个典型的软件定义数据中心的成功案例为SDN在数据中心的部署提供了巨大的参考价值。

然而当前的软件定义数据中心依然面临着巨大的挑战，主要表现为[16]：

(1) 资源管理的复杂性。数据中心作为资源密集型(如计算资源、网络资源和存储资源)基础设施需快速响应互联网用户的大量业务请求，并行处理多个具有海量数据特征的云计算任务。相比于传统局域网、广域网环境，云计算数据中心网络环境具有内部横向流量为主、服务器间带宽极高、业务流不可预测、端到端时延极低等特点。传统的经典广域网路由协议(如OSPFP协议等)在链路资源密集型(带宽大、数量多)数据中心的网络条件下存在不支持多路径传输、端到端传输速率受限、路由收敛速率缓慢等问题，难以满足数据中心内部流量传输的要求。同时，在广域网网上取得巨大成功的TCP协议，在云计算网络的多对一、多对多高速率数据传送的通信模式下的表现也不尽如人意，产生的TCP incast问题使得网络吞吐量极低，产生所谓的“吞吐量巧塌”现象。此外，基于泛洪的广播报文处理方式使以太网具有“即插即用”特性，这种处理方式占用大量交换机转发表空间和链路带宽、消耗终端CPU资源，与其配合使用的STP(spanning tree protocol)也同样存在诸多问题。

(2) SDN控制平面可扩展性。SDN控制平面的可扩展性是制约软件定义数据中心网络的主要性能瓶颈之一。在SDN中，网络交换节点的控制逻辑与数据转发在物理上进行分离，节点的控制能力全部集中到远程控制器上。虽然SDN的集中式架构使得控制器拥有全局网络信息，并通过开放网络可编程接口提高网络的灵活管控能力，但面对数据中心网络节点数目的不断增多、新业务流激增的情形，软件定义数据中心网络存在网络处理能力有限、控制消息处理时延过大，以及SDN控制平面可靠性差等问题。

(3) 软件稳定性问题。行业中一些Web巨头使用免费开源软件的能力非常惊人。但是这些超级公司拥有大量有支持开源软件的开发人员。对于没有足够资源或大型软件开发团队的企业而言，企业应用都必须建立在可靠且由供应商支持的解决方案上，同时他们也在密切关注开源解决方案，它们有可能逐渐成熟而变为可行方案。

(4) 安全性问题。软件定义的数据中心需要兼顾服务器、存储、网络等各个层面，并保证安全性。在通过云计算步入更开放、更具弹性的IT环境时，如何才能持续保持安全性是很多用户在云计算之路上最担心的问题。

参考文献

[1] 胡光武,徐恪,华婷,等.可编程路由器研究进展[J].中国教育网络,2010(7):40-43.

[2] 洪毅清,秦雅娟,周华春.基于NetFPGA的模块化硬件路由器实现[J].计算机应用与软件,2011,28(8):28-30.

[3] David Jacobs.协议之争:NETCONF足以和OpenFlow抗衡吗?[J/OL].http://www.searchnetworking.com.cn/showcontent_90818.htm.

[4] Amirthalingam K,Moorhead R J. SNMP-an overview of its merits and demerits[C]//Symposium on System Theory. IEEE,1995:180-183.

[5] Day M,Aggarwal S,Mohr G,et al. Instant Messaging/Presence Protocol Requirements[J]. Im Online! Ru Educause Review,2000,57(1):34-38.

[6] 高静,段会川.JSON数据传输效率研究[J].计算机工程与设计,2011,32(7):2267-2270.

[7] 李岩.json[J/OL]http://liyanblog.cn/articles/2012/09/28/1348813087862.html.

[8] Azarfar A,Frigon J F,Sanso B. Improving the Reliability of Wireless Networks Using Cognitive Radios[J]. Communications Surveys & Tutorials IEEE,2012,14(2):338-354.

[9] 徐珣,印骏.数据中心设计标准TIA-942解读[J].建筑电气,2009,28(11):17-19.

[10] 洪钊峰.历史与实现:不断演变的数据中心[J/OL].http://m.chinabyte.com/351/13036351__m.shtml

[11] 詹明非.软件定义存储技术及其应用研究[J].电信技术,2014(12):30-32.

[12] Gmbh I D. IDC benennt IBM als Nummer eins in Software-Defined-Storage-Marktreport[J].

[13] Sushant Jain,Alok Kumar,Subhasree Mandal,Joon Ong,Leon Poutievski,et al. B4: experience with a globally-deployed software defined wan. In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM (SIGCOMM '13),ACM,New York,NY,USA,2013:3-14.

[14] Hong CY,Kandula S,Mahajan R,Zhang M,Gill V,et al. Achieving high utilization with software-driven WAN. In:Proc. of the ACM SIGCOMM,2013:15-26.

[15] Helle B,Seetharaman S,Mahadevan P,Yiakoumis Y,Sharma P,Banerjee S,McKeown N. ElasticTree:Saving energy in data center networks. In:Proc of the USENIX NSDI,2010.

[16] 王健.基于软件定义网络架构的数据中心网络若干关键问题研究[D].北京:北京邮电大学信息与工程学院,2015.

第5章 软件定义网络

在第4章中已经提到基于控制与转发分离思想的可编程网络范式——软件定义网络(SDN)是近几年来受到广泛关注的一个技术理念。本章首先对SDN技术从定义、发展背景、整体架构方面进行介绍，然后分别对数据平面和控制平面的SDN关键技术进行讨论分析。

5.1 SDN技术简介

5.1.1 基本概念

对于当前的网络架构而言，SDN是一种新的实现方法，但它仍然属于一种网络概念。这里首先对相关的网络概念进行简单介绍，其中包括一些基本的传统网络概念与SDN网络模型下的相关名词。

1. OSI模型

OSI模型，即开放式通信系统互联参考模型(open system interconnection reference model，OSI/RM)，是国际标准化组织(ISO)提出的一个试图使各种计算机在世界范围内互连为网络的标准框架，简称OSI。OSI将计算机网络体系结构(architecture)划分为以下7层：

(1) 物理层——将数据转换为可通过物理介质传送的电子信号。

(2) 数据链路层——决定访问网络介质的方式，在此层将数据分帧，并处理流控制。本层指定拓扑结构并提供硬件寻址。

(3) 网络层——设计数据路由经过大型网络。

(4) 传输层——提供终端到终端的可靠连接。

(5) 会话层——允许用户使用简单易记的名称建立连接。

(6) 表示层——协商数据交换格式。

(7) 应用层——用户的应用程序和网络之间的接口。

因此，这个模型允许网络工程师和网络供应商将技术应用于特定的OSI层次。这种分割方式可以让工程师把某一网络应用的整个问

题分解到不同层次以方便管理。每一层都有特定的属性描述，每一层与相邻的层之间的交互方式都已经明确定义。

2. 交换机与路由器

交换机又称为交换式集线器，可以简单地理解为把一些计算机连接在一起组成一个局域网。交换机就是用来进行报文交换的机器。多为链路层设备(二层交换机)，能够进行地址学习，采用存储转发的形式来交换报文。交换机拥有一条很高带宽的背部总线和内部交换矩阵。交换机的所有端口都挂接在这条总线上，控制电路收到数据包以后，处理端口会查找内存中的地址对照表以确定目的 MAC(网卡的硬件地址)的 NIC(网卡)挂接在哪个端口上，通过内部交换矩阵迅速将数据包传送到目的端口，目的 MAC 若不存在则广播到所有的端口，接收端口回应后交换机会"学习"新的地址，并把它添加入内部 MAC 地址表中。

路由器又称网关设备，是用于连接多个逻辑上地分开的网络，逻辑网络代表一个单独的网络或者一个子网。当数据从一个子网传输到另一个子网时，可通过路由器的路由功能来完成。因此，路由器具有判断网络地址和选择 IP 路径的功能，它能在多网络互联环境中建立灵活的连接，可用完全不同的数据分组和介质访问方法连接各种子网，路由器只接受源站或其他路由器的信息，属网络层的一种互联设备，它的作用在于连接不同的网段并且找到网络中数据传输最合适的路径。

因此，交换机与路由器最明显的区别是工作层次不同，最初的交换机工作在数据链路层，而路由器设计在网络层。在 SDN 的网络概念中，它们都属于数据转发设备，了解交换机与路由器的基本概念，通过比较可以更好地理解 SDN 数据交换设备的工作方式。

3. 以太网

多个终端通过一个共享的基础设备连接到一起，形成一个广播域，主机之间使用 OSI 中第二层的介质访问控制(MAC)地址互相通信。

4. IP 地址和子网划分

IP 地址是指互联网协议地址，又译为网际协议地址。IP 地址是 IP 协议提供的一种统一的地址格式，它为互联网上的每一个网络和每一台主机分配一个逻辑地址，以此来屏蔽物理地址的差异。IP 地址是一个 32 位的二进制数，通常被分隔为 4 个"8 位二进制数"(也就是 4 个字节)。IP 地址通常用"点分十进制"表示成 a.b.c.d 的形式，其中，a、b、c、d 都是 0～255 之间的十进制整数。例如，点分十进 IP 地址 100.4.5.6 实际上是 32 位二进制数 01100100.00000100.00000101.00000110。

三类常用的 IP：

(1) A 类 IP 段——1.0.0.0～126.255.255.255 (0 段和 127 段不使用)。

(2) B 类 IP 段——128.0.0.0～191.255.255.255。

(3) C 类 IP 段——192.0.0.0～223.255.255.255。

XP 默认分配的子网掩码每段只有 255 或 0：

(1) A 类的默认子网掩码——255.0.0.0(一个子网最多可以容纳 1677 万多台计

算机)。

(2) B 类的默认子网掩码——255.255.0.0(一个子网最多可以容纳 6 万台计算机)。

(3) C 类的默认子网掩码——255.255.255.0(一个子网最多可以容纳 254 台计算机)。

必须有一个子网掩码,因为当配置 IP 时,所有计算机都必须填写子网掩码;必须在网络中设置一些逻辑边界;必须至少输入所使用 IP 类的默认子网掩码。子网的划分,实际上就是设计子网掩码的过程。子网掩码主要是用来区分 IP 地址中的网络 ID 和主机 ID,它用来屏蔽 IP 地址的一部分,从 IP 地址中分离出网络 ID 和主机 ID。子网掩码是由 4 个十进制数组成的数值,中间用“.”分隔。

5. 传输控制协议和用户数据报协议

OSI 模型第 4 层中的协议定义了网络上主机之间通信的方法。传输控制协议(TCP)使用面向连接的方式通信,而用户数据报协议(UDP)使用无连接模式通信。TCP 还具有流量控制、动态调整窗口/缓存大小以及显式确认等优点。

6. 网际控制报文协议

ICMP 是 Internet 控制报文协议。它是 TCP/IP 协议族的一个子协议,用于在 IP 主机、路由器之间传递控制消息。控制消息是指网络通不通、主机是否可达、路由是否可用等网络本身的消息。这些控制消息虽然并不传输用户数据,但是对于用户数据的传递起着重要的作用。

它可以帮助网络工程师排除网络故障和调整网络运作,某些平台上,ICMP 是 Ping 通信协议和路由跟踪的核心,并且可以用于 IP 网络主机之间通告传输错误和其他信息。

7. 北向接口

北向接口(northbound interface)是提供给其他厂家或运营商进行接入和管理的接口,即向上提供的接口。网络管理者必须使用北向接口定义和开发应用层中的网络管理应用程序,这样才能通过这些应用程序接入和管理网络,通常这种应用都是以简单、易用且直观的界面形式呈现给操作者,操作者通过界面单击或配置发送命令,而系统内部则使用北向接口将这些命令发送给数据处理层,数据处理层中驻留有随时运行着的北向接口处理进程,此进程接收到应用层发送来的命令后,将控制命令转发给下一层——数据管理层,以便继续执行。而接收到的如果是请求报文,则将来自数据管理层的数据按北向接口规定的格式封装后返回给应用层。JSON 和 Thrift 就是典型的北向接口。

8. 南向接口

南向接口(southbound interface)是与北向接口对应的一个概念,指管理其他厂家网管或设备的接口,即向下提供的接口。提供对其他厂家网元的管理功能,支持多种形式的接口协议,包括 SNMP、TR069、SYSLOG、SOAP、SSH 等接口,以及 I2RS、NETCONF 或者某些命令行操作界面。

在 SDN 中,OpenFlow 为典型的南向接口。

9. 东西向接口

东西向接口指在 SDN 架构中，用于多个控制器之间沟通和联系的接口。SDN 控制能力集中化，使得控制器的安全性和性能成为网络的最大限制，一旦控制器在性能或安全性上得不到保障，随之而来的是全网的服务能力的降级甚至是瘫痪。另外，单一的控制器也无法应对跨多个地域的 SDN 网络问题，需要多个 SDN 控制器组成的分布式集群，以避免单一的控制器节点在可靠性、扩展性、性能方面的问题。

10. 网络拓扑

网络拓扑是计算机网络各种网络组件（如链接、节点、接口、主机等）的分配布置方式。网络拓扑结构可以是物理上的，也可以是逻辑上的。物理拓扑结构是指网络的各种组件的位置，包括设备位置和线路的连接；而逻辑拓扑结构表示数据流在网络中的走向，与网络的实际物理结构无关。节点之间的物理连接方式和传输速率可能不同，两个网络之间的信号类型也可能不同，但是它们的拓扑结构却可能是相同的。

11. SDN 应用程序

SDN 应用程序（SDN application）更像是熟悉的网络上层功能，如 QoS、路由功能、Overlay 功能等。相比于传统网络，原本孤立的管理/配置如今被 SDN 统一化了，一个 APP 代表了整个 SDN 管理域内的所有此 APP 功能。例如网络出口要防 DDOS 攻击，调用了一个 APP 就能自动做黑洞引流操作，或者领导要开视频会议，调用一个 QoS 的 APP 就能全局做带宽质量保障，又如，通过 SDN 负载均衡 APP，可以实现根据不同业务/参数进行负载轮询[1]。

12. 应用程序接口

应用程序接口（API）是一组软件组件之间交互的编程规范。实际上，API 通常是一个包含变量规范、例程、对象类和数据结构的库。API 规范有国际标准（如 POSIX）、厂商技术文档（如 JunOS SDK）或编程语言的库文件等多种形式。

5.1.2 SDN 定义

随着 SDN 从学术界走到商业化，越来越多的开源项目出现，不同的标准化组织从各自的角度给出了相应的参考架构，同时各个运营商、网络公司、设备厂商均提出自己的 SDN 方案，因此至今业界对 SDN 的确切定义依然众说纷纭。

狭义上可以将 SDN 的定义等同于 OpenFlow，OpenFlow 起源于斯坦福大学的 Clean Slate 项目，该项目的最终目的是重新定义因特网架构，改变目前已经面临瓶颈且进化困难的网络基础架构，而 OpenFlow 协议将以太网的设计更一般化，将传统网络设备的数据转发（data plane）和路由控制（control plane）两个功能模块相分离，通过集中式的控制器（controller）以标准化的接口对各种网络设备进行管理和配置，为网络资源的设计、管理和使用提供更多的可能性，从而开创了 SDN 网络的革新与发展。从直观上来看，

OpenFlow 就是 SDN 概念在具体协议和技术上的体现形式。

从广义上，即网络架构层面定义 SDN，SDN 是一种数据控制分离、软件定义可编程的新型网络体系架构，开放网络基金会（open networking foundation，ONF）作为当前业界最活跃、规模最大的 SDN 标准组织，提出的 SDN 基本架构如图 5-1 所示。它具有集中式的控制平面和分布式的数据转发平面，两个平面互相分离，控制平面利用北向接口对转发平面上的网络设备进行集中式控制，并提供灵活的可编程能力，因此可以将广义 SDN 的特性概括为控制与转发分离、开放的编程接口、集中式的控制。

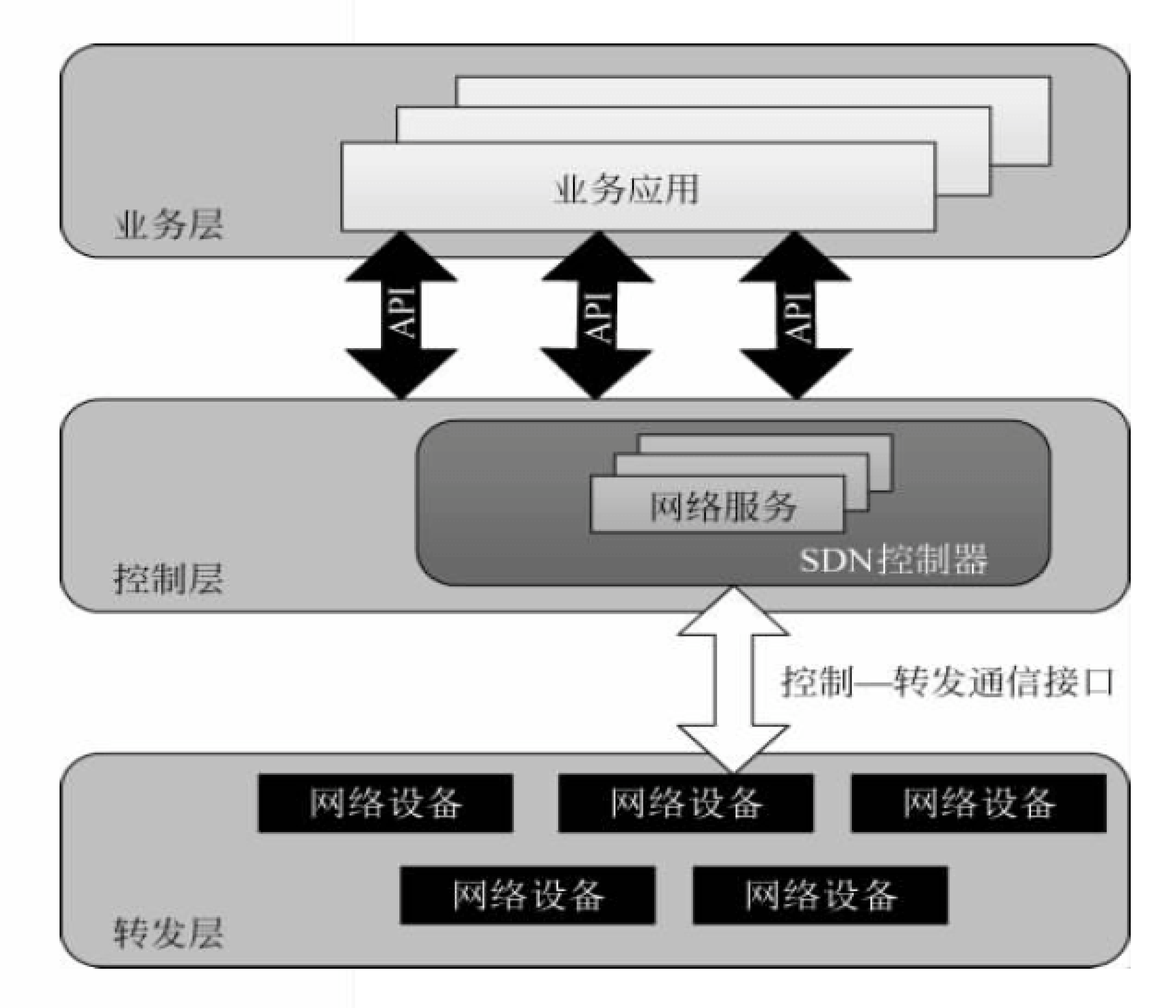


图 5-1　ONF 提出的 SDN 基本架构(引自参考文献[2])

在 SDN 架构中，控制平面通过控制—转发通信接口对网络设备进行集中控制，这部分控制信令的流量发生在控制器与网络设备之间，独立于终端间通信产生的数据流量，网络设备通过接收控制信令生成转发表，并据此决定数据流量的处理，不再需要使用复杂的分布式网络协议来进行数据转发，如图 5-2 所示。

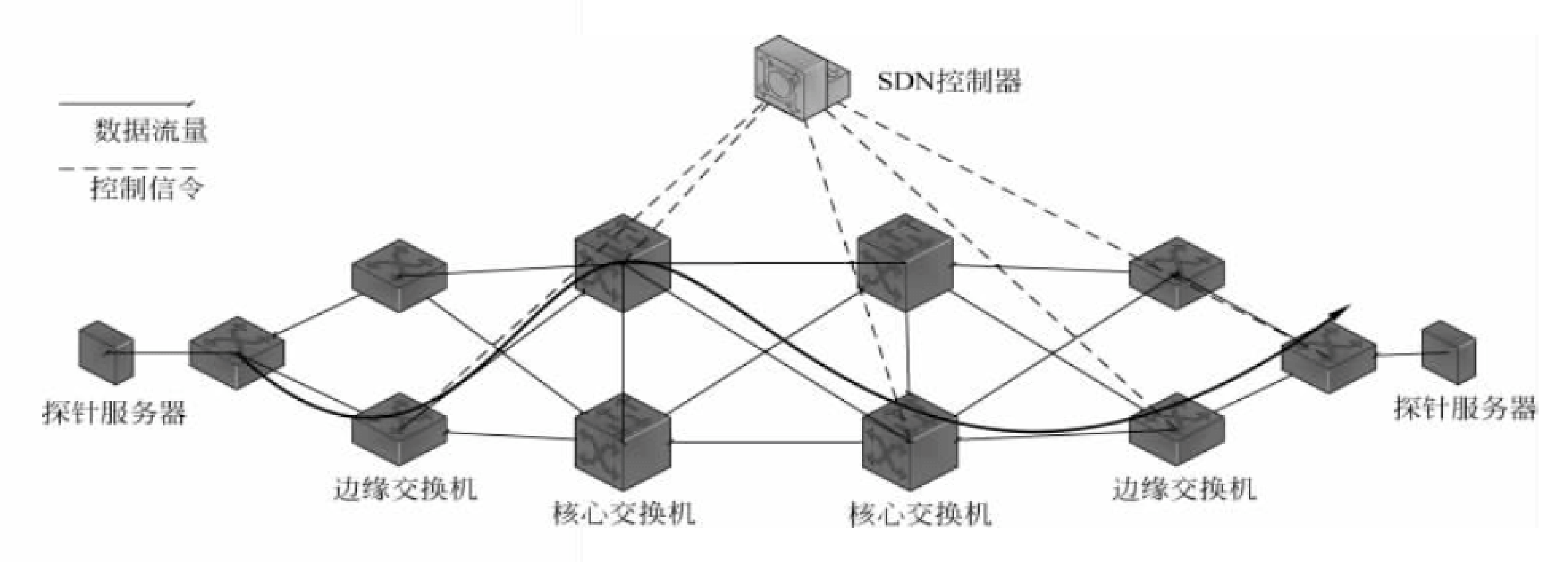


图 5-2　SDN 工作场景

（1）关于解耦合控制层面与数据转发层面。现有网络中，对流量的控制和转发都依赖于网络设备实现，且设备中集成了与业务特性紧耦合的操作系统和专用硬件，这些操作系统和专用硬件都是各个厂家自己开发和设计的。而在SDN网络中，网络设备只负责单纯的数据转发，可以采用通用的硬件；而原来负责控制的操作系统将提炼为独立的网络操作系统，负责对不同业务特性进行适配，而且网络操作系统和业务特性以及硬件设备之间的通信都可以通过编程实现。

（2）关于开放性的编程接口。SDN体系架构通过对整个网络进行抽象，为用户提供完备的编程接口，使用户可以根据上层的业务与应用个性化地定制网络资源来满足其特有的需求。由于其开放可编程的特性，SDN有可能打破某些厂商对设备、协议以及软件等方面的垄断，从而使更多人可以参与到网络技术的研发工作中来。

（3）关于集中控制的概念。对于传统的设备，因为不同的硬件、供应商私有的软件，使得网络本身相对封闭，只能通过标准的互通协议与计算设备配合运行。网络中所有设备的自身系统都是相对孤立和分散的，网络控制分布在所有设备中，网络变更复杂、工作量大，并且因为设备异构，管理上兼容性很差，不同设备的功能与配置差异极大；同时网络功能的修改或演进，会涉及全网的升级与更新。而在SDN的开放架构下，一定范围内的网络（或称SDN域），由集中统一的控制逻辑单元来实施管理，由此解决了网络中大量设备分散独立运行管理的问题，使得网络的设计、部署、运维、管理在一个控制点完成，而底层网络差异性也因为解耦合的架构得到了消除。集中控制在网络中引入了SDN区别于传统网络架构的角色——SDN控制器，也就是运行SDN网络操作系统并控制所有网络节点的控制单元。SDN能够提供网络应用的接口，在此基础上按照业务需求进行软件设计与编程，并且是在SDN控制器上加载，从而使得全网迅速升级新的网络功能，而不必再对每个网元节点进行独立操作。

综上所述，目前业界对SDN概念较为普遍的定义主要基于控制与转发分离、开放的编程接口与逻辑上的集中控制。随着SDN应用场景的复杂化，不同组织对SDN研究侧重点的多样化，SDN的定义在基于ONF的核心框架的基础上，也在不断完善补充的过程中。

5.1.3 SDN的发展背景

通过SDN的来源可以知道，SDN的初衷是改善目前网络遇到的瓶颈问题，然而，SDN具体是怎么改善网络的？与之前的网络改善途径有什么不同点？下面通过分析SDN的发展背景，并结合目前几种典型的商业SDN解决方案分析SDN的发展背景与应用前景。

传统互联网的主要问题在于配置复杂度高、研发周期长、架构封闭等问题，这些问题说明网络架构需要革新，可编程网络的相关研究为SDN的产生提供了可参考的理论依据。主动网络允许数据包携带用户程序，并能够由网络设备自动执行。用户可以通过编程方式动态地配置网络，达到了方便管理网络的目的。然而由于需求低、协议兼容性差等问题，并未在工业界实际部署。

针对当前网络的逻辑决策平面和分布式的硬件设备结合过紧的问题，Greenberg 等重新设计了互联网控制和管理结构，提出了如图 5-3 所示的 4D(decision，dissemination，discovery，data，决策，传播，发现，数据)体系架构。在 4D 架构下，决策平面通过全局网络视图做出网络控制决策，并直接下发到数据平面；传播平面在决策平面和路由器之间建立可靠的通信通道；发现平面负责发现网络中的物理组件，并为决策平面提供构建网络视图的基本信息；数据平面则实现数据转发功能。这种架构有助于实现健壮、安全的网络，便于对异构网络进行有效管理[3]。这一架构将可编程的决策平面(即控制层)从数据平面分离，使控制平面逻辑中心化与自动化，其设计思想产生 SDN 控制器的雏形[4]。

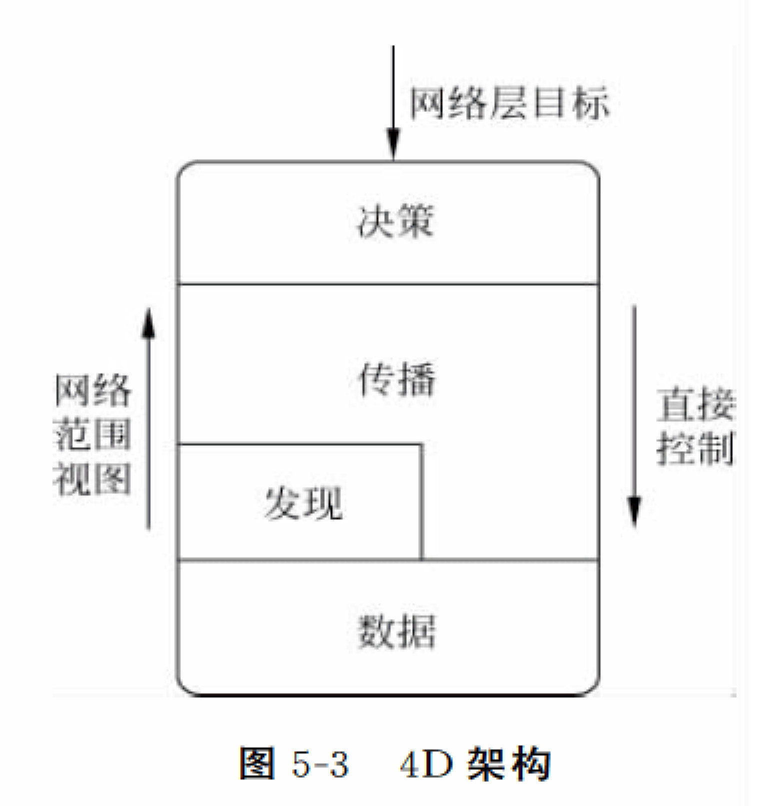


图 5-3　4D 架构

借鉴计算机系统的抽象结构，未来的网络结构将存在转发抽象、分布状态抽象和配置抽象这三类虚拟化概念[5]。转发抽象剥离了传统交换机的控制功能，将控制功能交由控制层来完成，并在数据层和控制层之间提供了标准接口，确保交换机完成识别转发数据的任务。控制层需要将设备的分布状态抽象成全网视图，以便众多应用能够通过全网信息进行网络的统一配置。配置抽象进一步简化了网络模型，用户仅需通过控制层提供的应用接口对网络进行简单配置，就可自动完成沿路径转发设备的统一部署。因此，网络抽象思想解耦了路径依赖，成为数据控制分离且接口统一架构(SDN)产生的决定因素。

SDN 接口的主流标准目前为 OpenFlow 协议，许多运营商和生产厂商根据该标准进行研发。互联网工程任务组(IETF)的 ForCES 工作组、互联网研究专门工作组(IRTF)的 SDNRG 研究组以及国际电信联盟远程通信标准化组织(ITU-T)的多个工作组同样针对 SDN 的新方法和新应用等展开研究。标准化组织的跟进是 SDN 市场快速发展的先决条件。据悉，SDN 市场已于 2013 年达到约 2 亿美元的产值，预计到 2016 年将达到 12 亿美元[6]，市场需求确保 SDN 有足够的发展空间。由此可见，SDN 具有广阔的发展前景和巨大的研究价值。下面从商业化方面，根据几个典型的 SDN 解决方案分析 SDN 带来的网络变革。

首先从性能上，可以以 SDN 的先期“试水者”谷歌为例，谷歌曾经部署过一个 SDN WAN 骨干，现在这个网络正在为其带来巨大的性能飞跃。事实证明，采用 SDN 可以获得更高的网络利用率。据了解，目前业界最佳的线路利用率也就在 30%～40%，而谷歌所运营的 SDN WAN 线路，只是通过细致的流量工程和优先策略，便可将利用率提升到接近 100%。换句话说，在出现故障时，可以牺牲没有任何严格交付期限的弹性流量，来保护高优先度的流量。

在网络简化方面，First Tracks 的首席分析师 Mark Leary 称，SDN 的初期收益就是对网络的简化。“围绕集中控制结构的整合可获得更高程度的自动化。这是一个可以立即看到影响的地方。渐进式的采用是成功的关键，”Mark Leary 说。“有些 SDN 解决方案的好处就在于它可以分成各自独立的部分。你可以把这些部分置入企业网络的有些

部分以便减少网络的复杂性，并可立刻看到收获。然后再进行扩展，最终简化企业的整个架构。SDN 的改进是动态的，例如可允许网络逐渐地适应负载的变化。”

从成本效益上看，另外一个典型的 SDN 成功案例，是印第安纳州立大学。该大学原本需要一台负载均衡设备放到 10Gb/s 互联网主干上去，把流量解析到多个入侵检测系统上去做分析，但市场反馈说这将花费 10～20 万美元去购买设备。这时他们才采用了 SDN，购买了一台价值仅 4 万美元、可支持 OpenFlow 的交换机去处理负载均衡任务，其成本效益是显而易见的。

“未来交换机将是白牌机的天下”，提到 SDN 的成本效益，许多人认为网络硬件的“白菜”成本将可能实现，低廉的白牌机将统治网络市场。按照“理想化”的 SDN 理念来说，这是一个不争的事实。但是，传统的网络市场被众多的巨头厂商所把持，这些厂商不会坐以待毙的，而事实也证明了这一点。目前，很多厂商都会在可用性、可管理性、性能、容量等方面极力实现差异化，因此 SDN 不太可能只是在构建网络时采用一些廉价的白牌机这么简单。

另外，SDN 使得产品升级周期更短，运营商们认为，SDN 能够高效使用通用服务器，从而帮助他们同时削减资本支出和运营成本，并为创造新服务提供一个更好的环境。此外，SDN 的关键益处还在于，它将使产品升级和更换周期变得更短、更容易，可以在通用的 X86 硬件上部署 SDN。然而，业内普遍也都承认，一些功能尚不能在这类服务器上进行部署，这在业界仍是一个富有争议的领域。

5.1.4 SDN 架构

目前各大厂商对 SDN 的系统架构都有自己的方案，针对不同的需求，各个组织也从不同的侧重点提出了许多相应的 SDN 参考架构。除了上述 ONF 提出的 SDN 基本模型，还有欧洲电信标准化组织（ETSI）针对运营商需求提出的 NFV 架构，得到了业界的广泛支持，以及各大设备厂商和软件公司提出的 OpenDaylight，作为一种 SDN 架构的具体实现形式已经部署在实际场景中。本节首先详细分析 ONF 提出的 SDN 体系结构中的组成部分，再与 NFV 和 OpenDaylight 的架构对比，以便读者对软件定义网络的架构以及相关项目有更明确的了解。

ONF 定义的 SDN 系统架构如图 5-4 所示，由下到上可以分为数据平面、控制平面和应用平面。该架构的最终目的是实现一套完整的编程接口开放给应用软件，这套接口称为 SDN 控制数据平面接口（control-data-plane in interface，CDPI）。控制平面上层的软件应用（SDN 应用程序）可以通过 CDPI 控制网络中的资源（数据层面设备资源）以及经过这些网络资源中的流量，并能根据应用需求灵活地调度这些流量。各个平面直接通过不同的协议进行交互，下面分别介绍一下该框架中的组成部分。

（1）数据平面：主要由网络元素构成，每个网元可以包含一个或者多个 SDN 数据路径，每个 SDN 数据路径是一个逻辑上的网络设备，数据平面没有控制功能，仅仅用于转发和处理数据。在逻辑上，数据平面代表全部或者部分物理资源。如图 5-4 所示，一个完整的网络元素包括 CDPI 代理、转发引擎或处理函数。

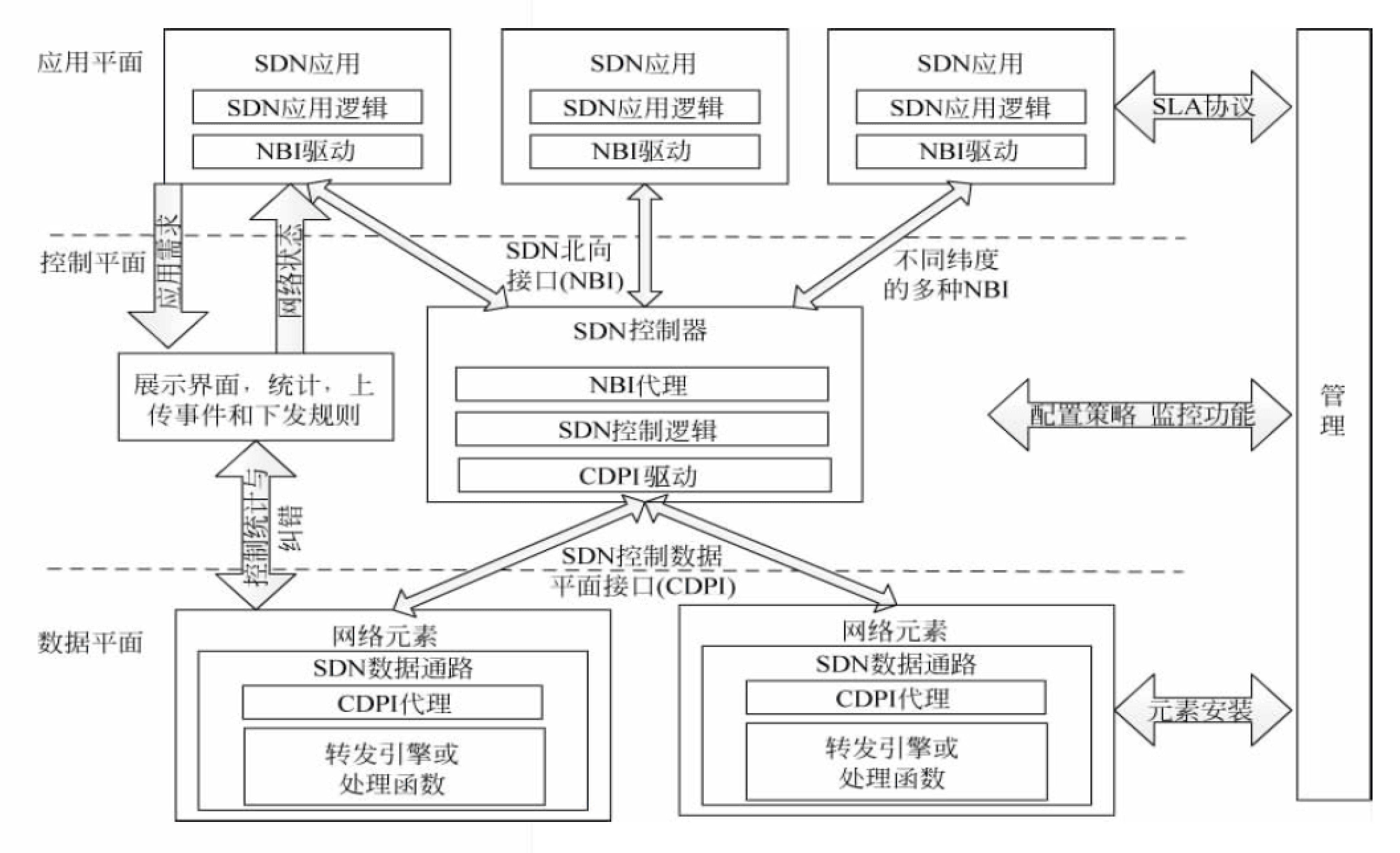


图 5-4　ONF 定义的 SDN 系统架构

（2）控制平面：控制平面将全网视图抽象成网络服务，通过访问 CDPI 代理来调用相应的网络数据通路，并为运营商、科研人员及第三方等提供易用的北向接口（NBI），方便这些人员订制私有化应用，实现对网络的逻辑管理。一般来说，SDN 控制器应该包括 CDPI 驱动、SDN 控制逻辑与 NBI 代理。

（3）应用平面：包含着各类基于 SDN 的网络应用，用户无须关心底层设备的技术细节，仅通过简单的编程就能实现新应用的快速部署。CDPI 负责将转发规则从网络操作系统发送到网络设备，它要求能够匹配不同厂商和型号的设备，而并不影响控制层及以上的逻辑。NBI 允许第三方开发个人网络管理软件和应用，为管理人员提供更多的选择。网络抽象特性允许用户可以根据需求选择不同的网络操作系统，而并不影响物理设备的正常运行。

再来看一下 NFV。NFV 是针对运营商网络出现的问题而提出的 SDN 解决方案。网络运营商的网络由专用设备来部署，随着各种新型网络服务的产生，这些专属设备功能变得繁杂，而管理这些繁杂的硬件设备造成运营成本及能耗的增加，从而导致运营商网络的发展遇到瓶颈。针对上述问题，NFV 将传统网络设备的软件与硬件相分离，使网络功能更新独立于硬件设备。为此，NFV 采用了资源虚拟化的方式，在硬件设备中建立一个网络虚拟层，负责将硬件资源虚拟化，形成虚拟计算资源、虚拟存储资源和虚拟网络资源等，运营商通过软件来管理这些虚拟资源。由于采用的是通用硬件设备，NFV 降低了设备成本，减少了能耗，缩短了新网络服务的部署周期，从而适应网络运营商的发展需求。NFV 架构在设计时参考了 ONF 的 SDN 架构设计，也体现了转发和控制层面的分离，控制层上提出了类似 SDN 中应用层的虚拟化架构的管理编排层，在接口设计方面，

NFV 既可以基于非 OpenFlow 协议,又能与 OpenFlow 协同工作,同时还支持多种传统接口标准化协议,以便适应网络运营商对设备的不同需求,并与 ONF 的 SDN 的发展保持相对独立。NFV 白皮书中提出的 NFV 架构如图 5-5 所示。

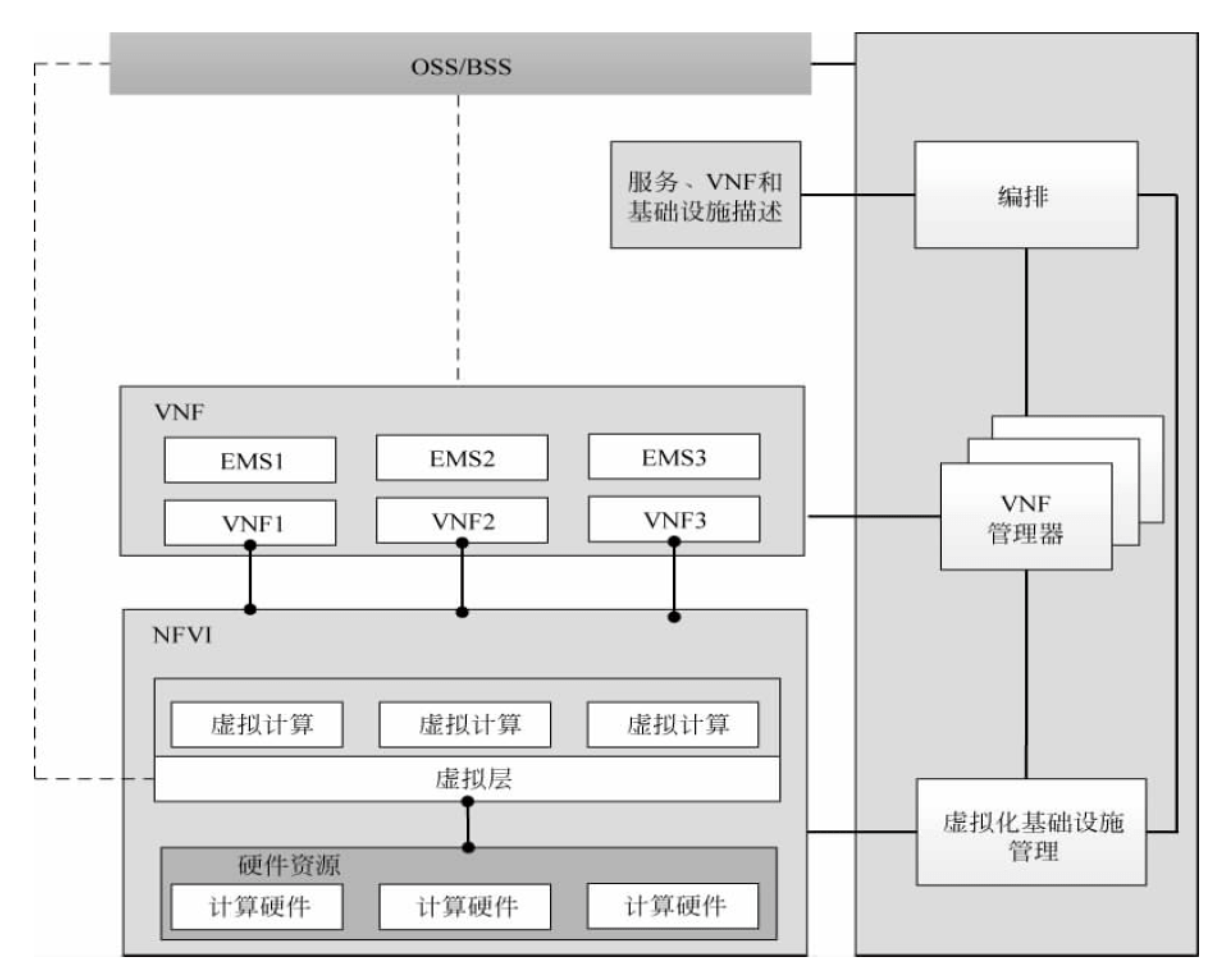


图 5-5 NFV 架构

OpenDaylight[7] 开源项目提出于 2013 年,主要参与者包括思科、Juniper 这类传统网络设备生产商、IBM、微软等传统 IT 软硬件设备厂商,以及 Arista、Big Switch 等新兴 IT 软件厂商。因此,OpenDaylight 考虑了兼容性问题,继承 SDN 架构形式的同时结合了 NFV 的特点。架构整体分为三个层次,分别是网络应用与业务流程(即应用层)、控制平台(即控制层)和物理与虚拟网络设备(即数据层)、OpenDaylight 的控制平台直接由自带的 Java 虚拟机实现。针对不同的网络任务,控制器自身携带了一系列可插入模块,并兼容第三方模块以增强 SDN 的功能。与 ONF 的 SDN 架构最大的不同在于:OpenDaylight 控制器的南向接口除了支持 OpenFlow 协议之外,还支持 NETCONF 等配置协议与 BGP 等路由协议,并支持生产厂商的专有协议(如思科的 OnePK 协议)。为了能够处理不同的标准协议,OpenDaylight 增加了服务抽象层 SAL,它负责将不同的底层协议标准转换成 OpenDaylight 控制层所理解的请求服务,保持了底层协议的透明性,并提高了整体架构的可扩展性。OpenDaylight 架构如图 5-6 所示。

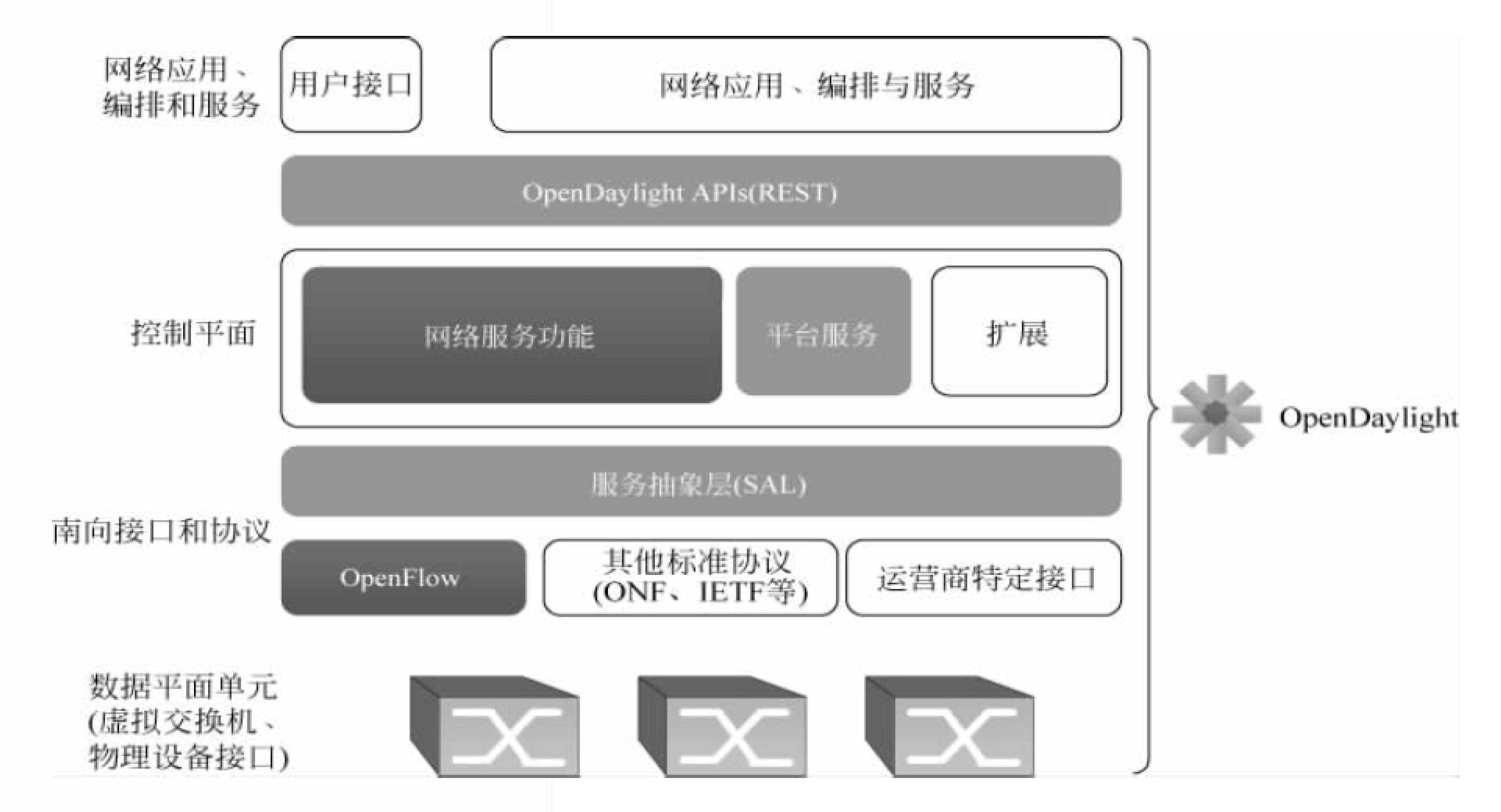


图 5-6 OpenDaylight 架构

综上所述，列出 SDN、NFV 和 OpenDaylight 的对比关系，如表 5-1 所示。在实际研究中，SDN 与 NFV 两种架构协同互补，将网络中的原有设备使用可编程设备代替，并结合控制层的改进促进网络的灵活性和可扩展性。OpenDaylight 由于其开源特性，可以兼容 SDN 与 NFV，以及未来与 SDN 并行的体系结构。

表 5-1 SDN、NFV 与 OpenDaylight 对比关系

结构	接口标准	SDN 兼容性	应用	发起机构	描述
SDN	OpenFlow	—	局域网、数据中心	科研人员	强调控制与转发分离
NFV	多种接口协议支持可扩展	可协同工作	运营商网络	运营商	强调网络工作虚拟化
OpenDaylight	多种接口协议支持可扩展	完全支持 OpenFlow	未来互联网	设备生产厂商和 IT 软件厂商	强调软件开源及实现

5.2 SDN 数据平面及南向接口技术

SDN 一直被提到的重要特点是数据转发与控制分离，数据转发设备是网络功能实现的基础，SDN 概念下以 OpenFlow 协议为代表的南向接口降低了相关设备的复杂度，允许 SDN 架构中的数据平面专注于高速转发数据分组，提高网络工作效率。针对这一变化带来的网络变革以及对网络设备商的影响，本节首先介绍网络数据平面的架构；为了避免交换机与控制器的频繁交互，双方约定基于流进行转发，该策略对灵活高效的南向接口协议提出需求，因此以 OpenFlow 为代表介绍 SDN 南向接口协议；最后，SDN 数据层的功能相对简单，技术研究主要面向交换机和转发规则方面，即如何设计可扩展的快

速转发设备，既可以灵活地匹配规则，又可以快速地转发数据，因此对 SDN 数据平面的软硬件设备进行介绍。

5.2.1 数据平面简介

在网络架构中，数据交换设备的主要工作都是实现数据信息在网络上的交换，这里的交换指的是一种技术概念，交换机的基本任务是处理和转发交换机各不同端口上各种类型的数据，即完成数据信息从设备入端口到出端口的转发。因此，只要是符合该定义的所有设备都可以被称为交换设备，L2/L3/ACL/QoS/组播/安全防护等各种具体的数据处理转发过程，都属于交换机数据平面的任务范畴。当描述数据网络第二层的设备时通常指的是交换机，当描述数据网络第三层设备的时候，通常指的是路由器或者三层交换机。

在国际标准化组织(ISO)定义的 7 层网络架构中，最初，数据面通过一系列链路层的操作来处理(通过有线、光网络或无线媒介等方式)传入的数据报，包括收集数据包，并执行基本检查链路层操作。在数据面中，通过一系列对传输数据的操作，使数据大小、对齐方式和封装要求符合协议标准，并对数据进行校验和完整性检查。特别地，一旦确定数据报的“类型”，可使用附加的规则检查特定的行为的类型。一个完整的，或者说正确的数据报，根据 MAC 或者 IP 地址等网络信息完成数据交换，随着设备功能的扩展与提升，也有一些结合了二层交换的简易与三层路由的智能特性的三层交换机。传统的功能架构由控制平面和数据平面组成，在物理上，它们是紧密耦合的，集成实现在单独的设备箱中。数据平面对每一个数据报文进行处理，使得它能够通过网络交换设备，这些设备大多数用专门的硬件实现，主要包括转发决策、背板和输出链路调度等功能。控制平面对交换机的转发表或者路由器的路由表进行管理，同时负责网络配置、系统管理等方面的操作，与转发平面相比，运行频率较低，可以采用软件实现。

在紧密耦合的传统网络架构中，各个设备的控制平面部署在网络节点上，无法对整体网络状态进行掌握，这就意味着无法根据全局状态对资源进行分配调度。另外，转发平面通常采用专门设计的 ASIC 芯片实现提升性能，控制平面则以控制软件或者网络操作的形式实现。传统网络架构中，网络设备供应商出于技术保密的考虑，基本不会开放相关接口供用户调用以进行网络的控制和管理，网络管理员很难获得网络的全网状态信息，为了实现网络配置，需要到现场对各个网络节点进行设置。

SDN 作为一种新型的网络范例，打破了传统网络垂直组网模式，将网络控制逻辑从基础层的路由器和交换机中分离出来，也就是在数据平面“去掉”控制功能，在控制平面实现了可编程网络，更加高效地路由和更加高效地使用链路资源。同时这样的网络更容易创建并引入网络抽象，简化了网络管理并促进网络演进。

在 SDN 的架构中提到，数据平面主要由网络元素构成，每个网元可以包含一个或者多个 SDN 数据路径，每个 SDN 数据路径是一个逻辑上的网络设备，在逻辑上，数据平面代表全部或者部分物理资源。这里的“网络元素”可以认为是具有基本数据转发功能的网络设备，即 SDN 交换机。为了在不涉及网络管理功能的状态下保证数据转发效率，OpenFlow 提出以“流”的方式转发数据，并以定义“流”的方式实现控制平面与数据平面

的交互，因此，可以认为，负责 SDN 网络中数据转发的 SDN 交换机设备和以 OpenFlow 为代表的南向接口协议构成了 SDN 架构中的数据平面。本节通过对这两方面的介绍分析，以更加直观地展现 SDN 数据平面的工作方式与特点。

5.2.2 OpenFlow 协议简介

1. OpenFlow 基本概念

OpenFlow 最初是由斯坦福大学网络研究构想的一部分，出自 Clean Slate 计划资助的 Ethane 项目，该项目致力于企业网架构的更新，而提出 OpenFlow 的目的是在校园网中创建一个可以供实验者运行试验协议的平台，脱离运营商的限制，以进行研究和试验。在 OpenFlow 提出之前，为了进行网络实验，要求设备制造商完全开放平台接口或实验者自行制造设备。设备制造商完全开放平台接口让研究者可以使用商用网络设备进行二次开发，寻找实验协议与传统协议并存的方法。但是，直接开放网络设备的开发接口对设备提供商而言是一场噩梦，一方面与商用平台的封闭性相冲突，开放开发的二次接口无疑会有暴露自身技术细节的风险，为竞争对手提供了机会，或者为新兴厂家提供了进入行业的门槛。而实验者自行制造设备，大大延长了研发周期与代价。

Nick McKeown 等于 2008 年在 ACM SIGCOMM 发表了题为 OpenFlow：Enabling Innovation in Campus Networks 的论文，首次详细地介绍了 OpenFlow 的概念。该篇论文除了阐述 OpenFlow 的工作原理外，还列举了 OpenFlow 几大应用场景，包括：①校园网络中对实验性通信协议的支持（如其标题所示）；②网络管理和访问控制；③网络隔离和 VLAN；④基于 Wi-Fi 的移动网络；⑤非 IP 网络；⑥基于网络包的处理。当然，目前关于 OpenFlow 的研究已经远远超出了这些领域。该论文表现了 OpenFlow 的原理和基本架构，OpenFlow 规范主要定义了 Switch 的功能模块以及其与控制器之间的通信信道等方面。在控制器与交换机之间有一条“安全通道”，安全通道是从控制器到交换机之间的接口，在 OpenFlow v1.0 中规定该安全通道需要使用 TLS 安全隧道。而从 v1.1 开始，OpenFlow 不再强制要求使用 TLS 隧道，而是可以使用普通的 TCP 连接。另外，在各个 OpenFlow 版本中，均建议在缺省情况下使用 TCP 6633 端口用于安全通道。

要了解 OpenFlow，要首先了解“流”（flow）的概念。IP 网络是基于数据分组转发的分组交换网络，一次网络通信会产生大量的数据分组，虽然数据组之间具有联系，但是传统网络却对它们进行单独处理，造成了网络处理效率低下。如果能通过提取这次通信产生的数据分组的共同特征，如 MAC 地址、IP 地址等，将其抽象为“流”，根据网络设备对这些数据分组进行处理，可以有效提高网络设备对数据组的处理效率。

而“流表”（FlowTable）是对于网络设备的数据转发功能的一种抽象。OpenFlow 通过用户定义的或者预设的规则来匹配和处理网络包。一条 OpenFlow 的规则由匹配域（match fields）、优先级（priority）、处理指令（instructions）和统计数据（如 counters）等字段组成，在一条规则中，可以根据网络包在 L2、L3 或者 L4 等网络报文头的任意字段进行匹配，如以太网帧的源 MAC 地址、IP 包的协议类型和 IP 地址，或者 TCP/UDP 的端口号等。目前 OpenFlow 的规范中还规定了 Switch 设备厂商可以选择性地支持通配符进

行匹配。据说，OpenFlow 在未来还计划支持对整个数据包的任意字段进行匹配。所有 OpenFlow 的规则都被组织在不同的 FlowTable 中，在同一个 FlowTable 中按规则的优先级进行先后匹配。一个 OpenFlow 的 Switch 可以包含一个或者多个 FlowTable，从 0 依次编号排列。OpenFlow 规范中定义了流水线式的处理流程。当数据包进入 Switch 后，必须从 FlowTable 0 开始依次匹配；FlowTable 可以按次序从小到大越级跳转，但不能从某一 FlowTable 向前跳转至编号更小的 FlowTable。当数据包成功匹配一条规则后，将首先更新该规则对应的统计数据（如成功匹配数据包总数目和总字节数等），然后根据规则中的指令进行相应操作，如跳转至后续某一 FlowTable 继续处理，修改或者立即执行该数据包对应的 Action Set 等。当数据包已经处于最后一个 FlowTable 时，其对应的 Action Set 中的所有动作将被执行，包括转发至某一端口，修改数据包某一字段，丢弃数据包等。OpenFlow 规范中对目前所支持的处理指令和动作进行了完整详细的说明和定义。

上面已经提到数据层面的网络设备，现在看一下网络设备的 OpenFlow 端口（port）。OpenFlow v1.2 及其后续版本提出了 OpenFlow 端口的概念。OpenFlow 的端口是 OpenFlow 进程和网络之间传递数据包的网络接口。OpenFlow 交换机之间通过 OpenFlow 端口在逻辑上相互连接。因此，OpenFlow 交换机必须支持三种类型的 OpenFlow 端口：

（1）物理端口。OpenFlow 的物理端口为交换机定义的端口，与 OpenFlow 交换机上的硬件接口一一对应。在某些部署中，OpenFlow 交换机可以实现交换机的硬件虚拟化。在此情况下，一个 OpenFlow 物理端口可以对应交换机硬件接口的一个虚拟接口。

（2）逻辑端口。OpenFlow 的逻辑端口为交换机定义的端口，但并不直接对应一个交换机的硬件接口。逻辑端口是更高层次的抽象概念，可以是交换机中定义的其他一些端口（如链路聚合组、隧道、环回接口等）。逻辑端口可能支持报文封装并被映射到不同的物理端口上，但其处理动作必须是透明的，即 OpenFlow 在处理上并不刻意区分逻辑端口和物理端口的差异。物理端口和逻辑端口之间的唯一区别是：一个逻辑端口的数据包可能增加了一个额外的元数据字段即隧道 ID，而当一个逻辑端口上接收到的报文被发送到控制器时，其逻辑端口和底层的物理端口都要报告给控制器。

（3）保留端口。OpenFlow 保留端口用于特定的转发动作，如发送到控制器、洪泛，或使用非 OpenFlow 的方法转发，如使用传统交换机的处理过程。

无论物理端口、逻辑端口还是保留端口，它们都被称作 OpenFlow 标准端口。这些端口可以被用作 OpenFlow 交换机的入端口和出端口，也可用于组表之中，同时每个端口都有相应的端口计数器。

2. OpenFlow 协议分析

1）数据包匹配

下面以 OpenFlow1.0 为例分析，OpenFlow 流表包含数据包匹配特征和数据包处理方法。首先来看数据包匹配特征：

（1）一层——交换机入端口（ingress port）。

（2）二层——源 MAC 地址（Ether source）、目的 MAC 地址（Ether dst）、以太网类型

(EtherType)、VLAN 标签(VLAN id)、VLAN 优先级(VLAN priority)。

(3) 三层——源 IP(IP src)、目的 IP(IP dst)、IP 协议字段(IP proto)、IP 服务类型(IP ToS bits)。

(4) 四层——TCP/UDP 源端口号(TCP/UDP src port)、TCP/UDP 目的端口号(TCP/UDP dst port)。

综上所述,包头域中用于和交换机接收到的数据包进行匹配的元组涵盖了 ISO 网络模型中第二至第四层的网络配置信息。每一个元组中的数值可以是一个确定的值或者是 ANY 以支持对任意值的匹配。另外,如果交换机能够在 IP 地址相关元组上支持子网掩码,则将有助于实现更精确的匹配。

2) 数据包处理

数据包处理方法有转发(forward)和修改包头(modify field),数据包转发行为可以分为对物理端口和虚拟端口的转发,如表 5-2 和表 5-3 所示。

表 5-2 数据包转发处理

数据包处理	端口	转发行为	说　明
转发行为	物理端口	ALL	转发给所有出端口,但不包括入端口
		CONTROLLER	将数据包发送给控制器
		LOCAL	将数据包发送给交换机本地端口
		TABLE	将数据包按照流表匹配条目处理
		IN_PORT	将数据包从入端口发出
	虚拟端口	NORMAL	按照普通二层交换机流程处理数据包
		FLOOD	将数据包从最小生成树使能端口转发(不包括入端口)

表 5-3 数据包的修改包头行为

数据包处理	修改行为	说　明
修改包头	SET_VLAN_VID	修改 VLAN 标签
	SET_VLAN_PCP	修改 VLAN 优先级
	STRIP_VLAN	弹出 VLAN 标签
	SET_DL_SRC	修改源 MAC 地址
	SET_DL_DST	修改目的 MAC 地址
	SET_NW_SRC	修改源 IP 地址
	SET_NW_DST	修改目的 IP 地址
	SET_NW_TOS	修改 IP 服务类型字段
	SET_TP_SRC	修改源端口号
	SET_TP_DST	修改目的端口号

表 5-2 和表 5-3 中所示每一种操作称为一个动作(action),流表中的数据包处理方法是一个动作列表(action list),动作列表由以上各种动作组合合成。OpenFlow 交换机的每个流表项可以对应有零至多个动作,如果没有定义转发动作,那么与流表项包头域匹配的数据包将被默认丢弃。同一流表项中的多个动作的执行可以具有优先级,但是在数

据包的发送上并不保证其顺序。另外，如果流表项中出现有 OpenFlow 交换机不支持的参数值，交换机将向控制器返回相应的出错信息。

OpenFlow 交换机在接收到网络数据包后，对其开展的处理流程如图 5-7 所示。

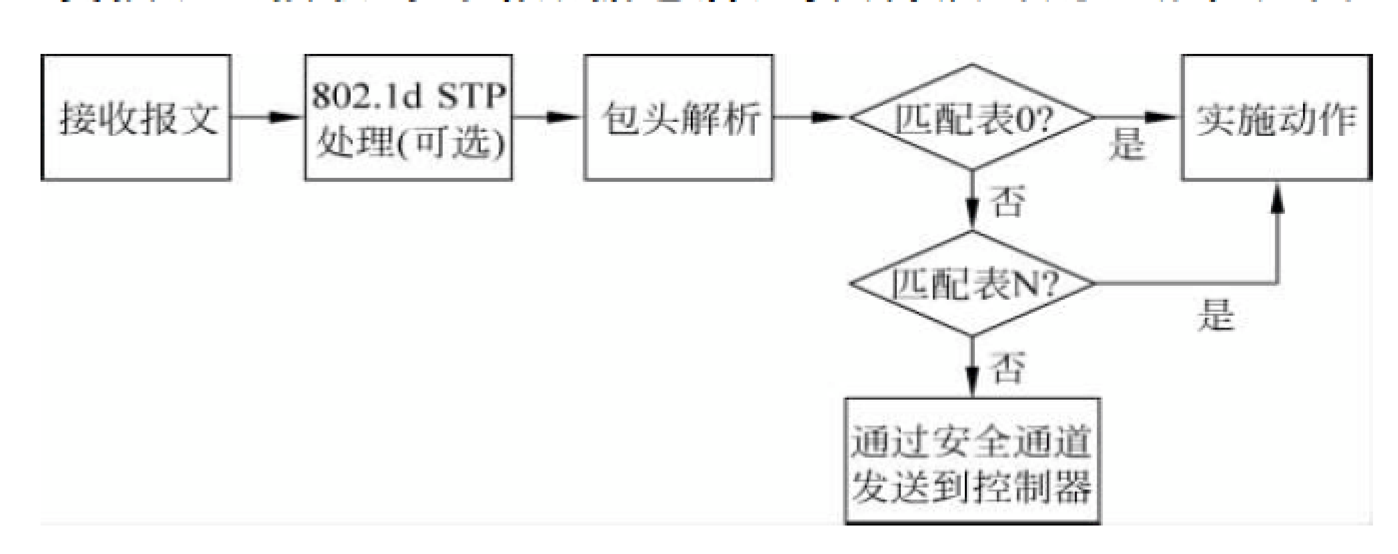


图 5-7　OpenFlow 交换机中的数据包处理流程

流程中对 802.1d 协议的处理是流程中的可选步骤（在 OpenFlow 1.1 之后已删除）。当 OpenFlow 交换机接收到一个数据包时，将按照优先级依次匹配其本地保存的流表中的表项，并以发生具有最高优先级的匹配表项作为匹配结果，还将根据相应的动作对数据包进行操作。同时，一旦匹配成功，对应的计数器将更新；而如果没能找到匹配的表项，则将数据包转发给控制器。

OpenFlow 交换机对数据包头的解析和匹配流程如图 5-8 所示。交换机中每一个表项的匹配首先按照接收到数据包的物理端口对入端口进行匹配，然后按照二层数据包头进行比较。如果以太网类型为 0x8100，即数据包是 VLAN 包，则继续查询 VLAN ID 和 PCP 域。如果以太网类型为 0x0806，则为 ARP 包，继续查询源 IP 地址和目的 IP 地址。如果以太网类型为 0x0800，即为 IP 包，则继续查询 IP 包头的相关域。如果 IP 包是 TCP/UDP 包，则还需继续查询传输层端口。如果 IP 包是 ICMP 包，则继续查询 ICMP 包中的 Type 和 Code。对于分段数据包的后续包，则将传输层端口设为 0 后继续查询。

3）OpenFlow 消息

协议由三种消息组成：控制器—交换机消息（controller-to-switch）、异步消息（asynchronous）和对称消息（symmetric）。

（1）控制器—交换机消息由控制器发送，可以指定修改或删除流定义、请求交换机功能信息、查询交换机信息（如计数器），同时可以在新流创建之后，将数据包发送回交换机。

（2）异步消息由交换机发送，可以向控制器发送与现有流不匹配的数据包，在流的存活参数时间已到或静止计时器过期时，通知控制器流已经删除、通知控制器端口状态变化或交换机出现错误。

（3）对称消息可以由交换机或控制器双方发出，可用于在控制器和交换机之间启动时发送握手消息、发送回波消息、测试控制器与交换机之间的连接延迟，确定控制器与交换机的连接是否仍然有效发送试验消息，作为将来 OpenFlow 技术的扩展方法。

各类消息的细节如表 5-4 所示。

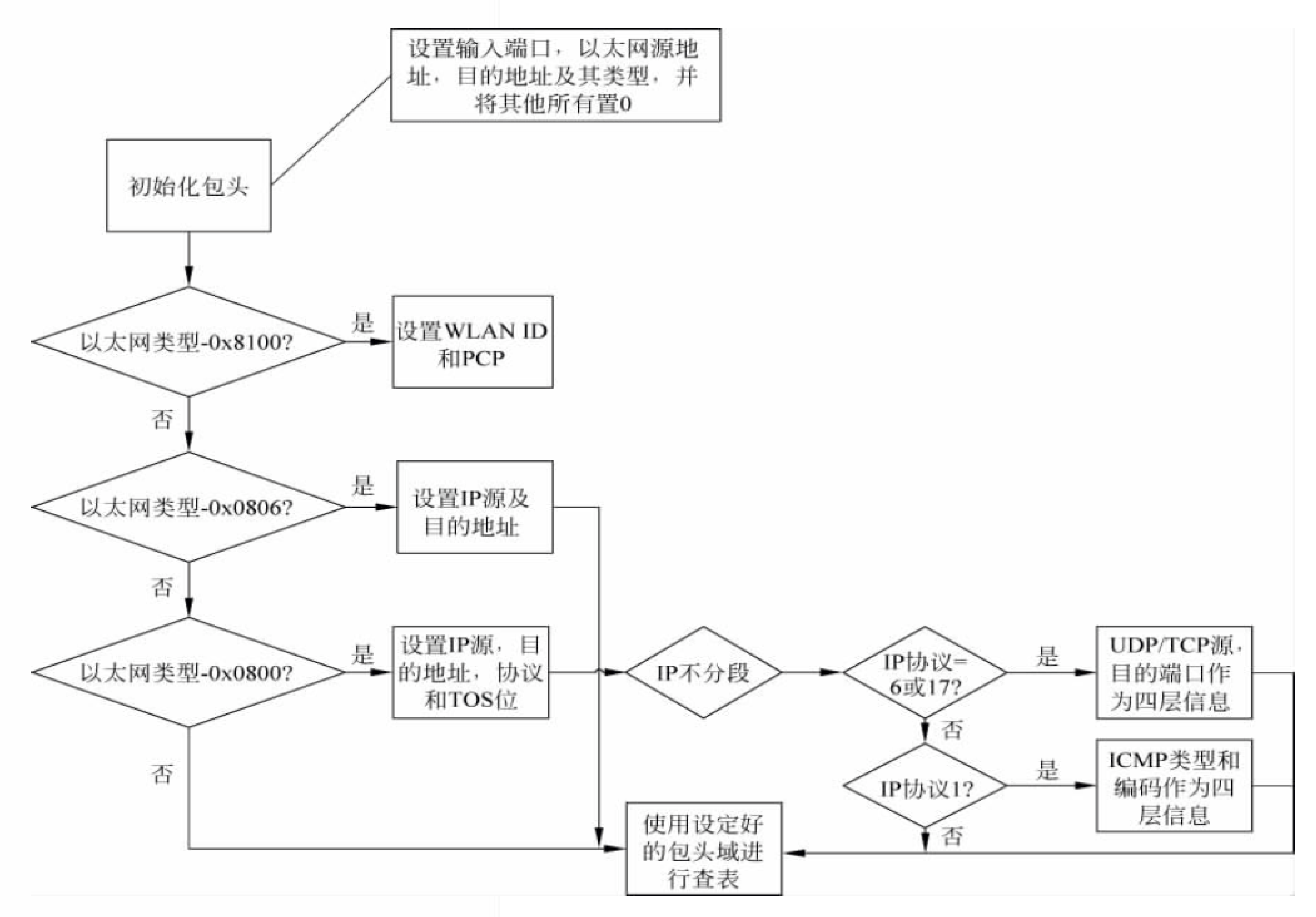


图 5-8　OpenFlow 交换机中数据包头解析和匹配流程

表 5-4　OpenFlow 协议消息

类　　型	名　称	说　　明	备　　注
控制器—交换机消息	Features	用来获取交换机特性	控制器至交换机消息，此类消息由控制器主动发出。OpenFlow 交换机接收并处理可能发送或不需要发送的应答消息
	Configuration	用来配置 OpenFlow 交换机	
	Modify-State	用来修改交换机状态（修改流表）	
	Read-State	用来读取交换机状态	
	Send-Packet	用来发送数据包	
	Barrier	阻塞消息	
异步消息	Packet-in	用来告知控制器：交换机接收到数据包	由 OpenFlow 交换机主动发起，用来通知交换机上发生的某些异步事件。 消息是单向的，不需要控制器应答。主要用于交换机向控制器通知收到报文、状态变化及出现错误等事件信息
	Flow-Removed	用来告知控制器：交换机流表被删除	
	Port-Status	用来告知控制器：交换机端口状态更新	
	Error	用来告知控制器：交换机发生错误	
对称消息	Hello	用来建立 OpenFlow 连接	本类消息控制器和交换机都可以主动发起，并需要接收方应答。这些都是双向对称的消息，主要用来建立连接、检测对方是否在线等
	Echo	用来确认交换机与控制器之间的连接状态	
	Vendor	厂商自定义消息	

图 5-9 展示了 OpenFlow 和交换机之间一次典型的消息交换过程。出于安全和高可用性等方面的考虑，OpenFlow 的规范还规定了如何为控制器和交换机之间的信道加密、如何建立多连接等（主连接和辅助连接）。

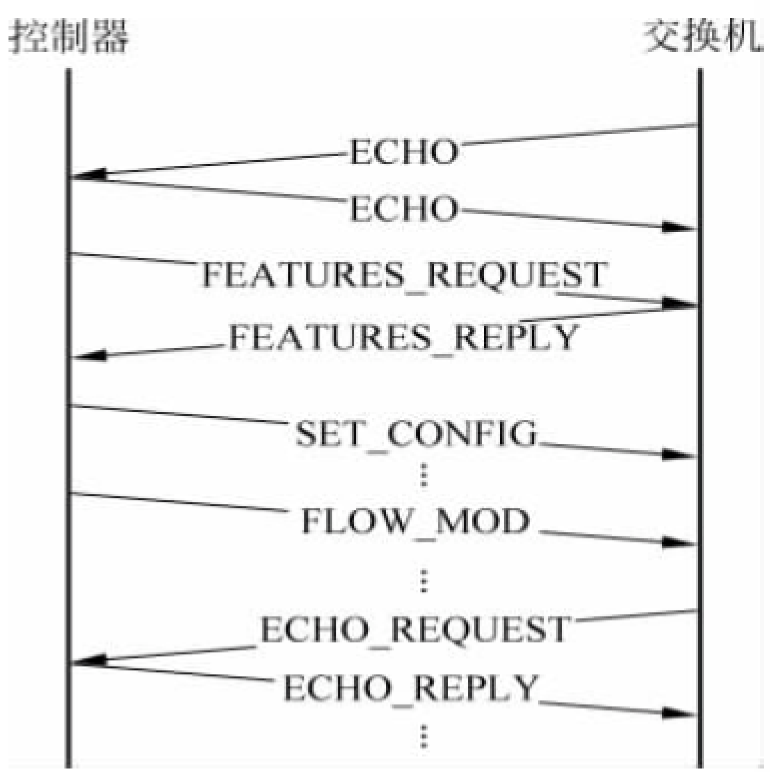


图 5-9 OpenFlow 消息交换过程

3. OpenFlow 演进

自 2009 年底发布第一个正式版本 v1.0 以来，OpenFlow 协议已经经历了 1.1、1.2、1.3 以及最新发布的 1.4 等版本的演进过程。同时，2012 年 OpenFlow 管理和配置协议也发布了第一个版本（OF-CONFIG1.0&1.1），用于配合 OpenFlow 协议进行自动化的网络部署。图 5-10 给出了 OpenFlow 协议各个版本的演进过程和主要变化。目前使用和支持最多的是 OpenFlow1.0 和 OpenFlow1.3 版本。图 5-11 给出了 OpenFlow1.3 版本交换机架构图，可以将其与图 5-7 进行对比。

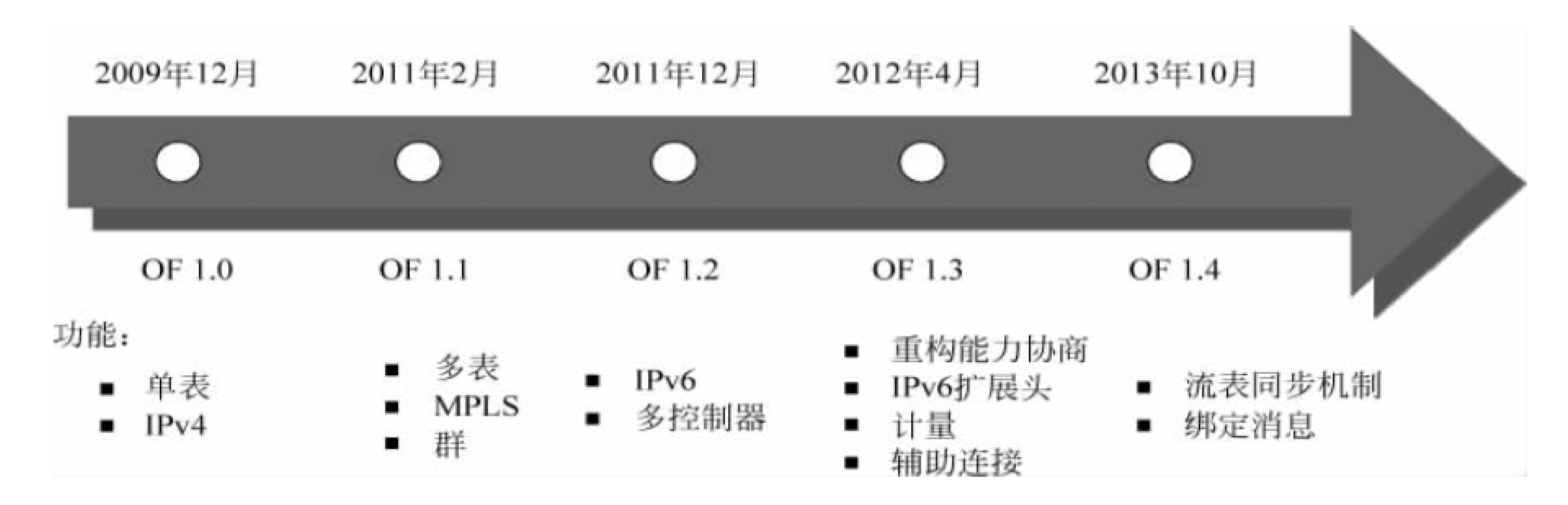


图 5-10 OpenFlow 协议版本演进图

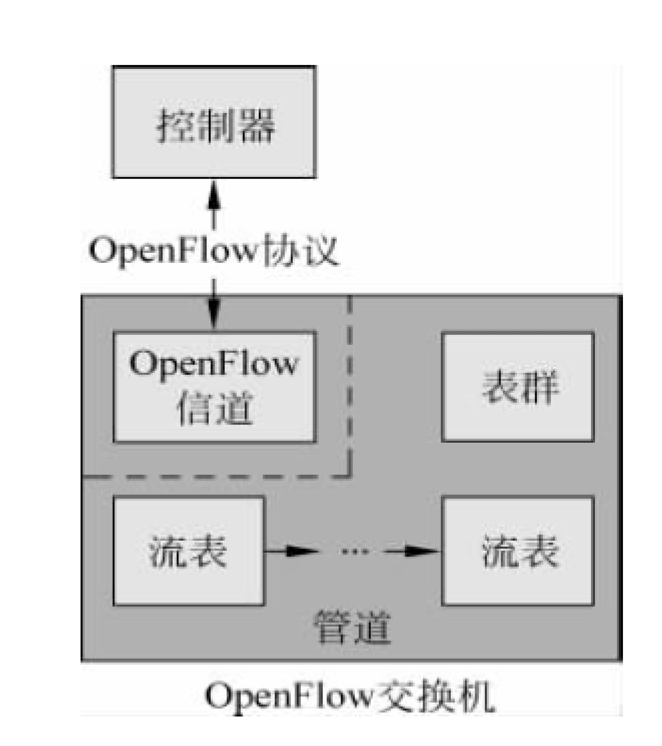


图 5-11 OpenFlow 1.3 版本交换机架构图

OpenFlow1.0 协议指定每个 OpenFlow 交换机中都存在一张流表[8]，用于数据包查找、处理和转发，并且只能同一台控制器进行通信，流表的维护也是通过控制器下发相应的 OpenFlow 消息来实现。流表由多个流表项组成，而每个流表项就是一个转发规则。流表项由匹配字段、计数器和动作组成。其中，匹配字段是流表项的标识，OpenFlow1.0 支持 12 个匹配字段；计数器用于流表项的匹配和收发包统计；动作指示对匹配流表项的数据包应该执行的动作，如转发到另一端口，丢弃或送控制器处理，甚至可以修改数据包字段转发。但 OpenFlow1.0 只支持 IPv4。OpenFlow1.0 版本的优势是它可以与现有的商业交换芯片兼容，

通过在传统交换机上升级固件就可以支持 OpenFlow1.0 版本，既方便 OpenFlow 的推广使用也有效保护了用户的投资，因此 OpenFlow1.0 是目前使用和支持最广泛的协议版本。

自 OpenFlow1.1 版本开始支持多级流表，将流表匹配过程分解成多个步骤，形成流水线处理方式，这样可以有效和灵活利用硬件内部固有的多表特性，同时把数据包处理流程分解到不同的流表中也避免了单流表过度膨胀问题。除此之外，OpenFlow1.1 中还增加了对于 VLAN 和 MPLS 标签的处理，并且增加了 Group 表，通过在不同流表项动作中引用相同的组表实现对数据包执行相同的动作，简化了流表的维护。OpenFlow1.1 版本是 OpenFlow 协议版本发展的一个分水岭，它和 OpenFlow1.0 版本开始不兼容，但后续版本仍然还是在此基础上发展。

为了更好地支持协议的可扩展性，OpenFlow1.2 版本发展为下发规则的匹配字段不再通过固定长度的结构来定义，而是采用了 TLV 结构定义匹配字段，称为 OXM (OpenFlow extensible match)，这样用户就可以灵活地下发自己的匹配字段，增加了更多关键字匹配字段的同时也节省了流表空间。同时，OpenFlow1.2 规定可以使用多台控制器和同一台交换机进行连接增加可靠性，并且多控制器可以通过发送消息来变换自己的角色。还有重要的一点是，自 OpenFlow1.2 版本开始支持 IPv6。

经过 1.1 和 1.2 版本的演变积累，2012 年 4 月发布的 OpenFlow1.3 版本成为长期支持的稳定版本。OpenFlow1.3 流表支持的匹配关键字已经增加到 40 个，足以满足现有网络应用的需要。OpenFlow1.3 主要还增加了计量表，用于控制关联流表的数据包的传送速率，但控制方式目前还相对简单。OpenFlow1.3 还改进了版本协商过程，允许交换机和控制器根据自己的能力协商支持的 OpenFlow 协议版本。同时，连接建立也增加了辅助连接提高交换机的处理效率和实现应用的并行性。其他还有 IPv6 扩展头和 Table-miss 表项的支持。

2013 年最新发布的 OpenFlow1.4 版本仍然是基于 1.3 版本的特征改进版本，数据转发层面没有太大变化，主要是增加了一种流表同步机制，多个流表可以共享相同的匹配字段，但可以定义不同的动作；另外又增加了绑定消息，确保控制器下发一组完整消息或同时向多个交换机下发消息的状态一致性。其他还支持端口属性描述、多控制器相关的流表监控等特征。

OpenFlow 协议的发展演进一直都围绕着两个方面，一方面是控制面增强，让系统功能更丰富、更灵活；另一方面是转发层面的增强，可以匹配更多的关键字，执行更多的动作。每一个后续版本的 OpenFlow 协议都在前一版本的基础上进行了或多或少的改进，但自 OpenFlow1.1 版本开始和之前版本不兼容，OpenFlow 协议官方维护组织 ONF 为了保证产业界有一个稳定发展的平台，把 OpenFlow1.0 和 1.3 版本作为长期支持的稳定版本，一段时间内后续版本发展要保持和稳定版本的兼容。

5.2.3 OF-CONFIG 协议简介

在 OpenFlow 协议中，有控制器向 OF 交换机发送流表以控制数据流的转发行为，但是没有规定如何去管理和配置这些 OF 交换机，而 OF-CONFIG 就是为了解决这一问题

而提出的。OF-CONFIG 协议最初被设计用于在网络组件中 OpenFlow 相关信息的设定(OF-CONFIG 1.0)。这个协议是依靠 XML 架构、Yang data 模型和用于交付的 NETCONF 协议建立的。在 1.1 版的 OF-CONFIG 中,标准将自身与运营商运行 FlowVisor 分离开来[或者类似的外部分层代理(external slicing proxy)]以在物理交换机上实现多个虚拟交换机。这会改变交换机的工作模式,物理交换机可以有多个内部逻辑交换机,如图 5-12 所示。

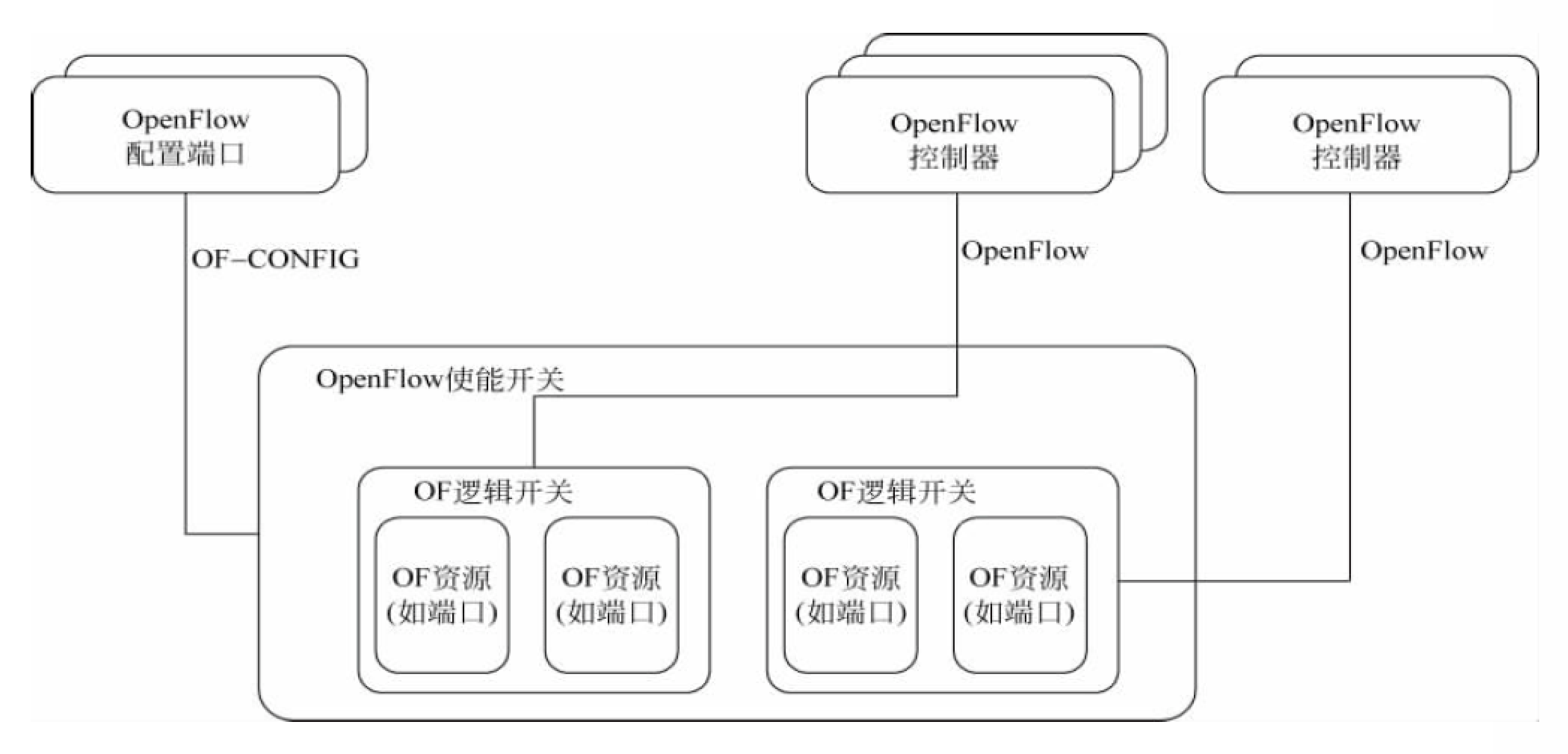


图 5-12 OF-CONFIG 与线路协议的关系

如图 5-12 所示,OF-CONFIG 在 OpenFlow 架构上增加了一个被称作 OpenFlow 配置点的配置节点。这个节点本质上是运行着 OF-CONFIG 客户端进程的计算机,它既可以是控制器上的一个软件进程,也可以是传统的网管设备,它通过 OF-CONFIG 协议对 OpenFlow 交换机进行管理,因此 OF-CONFIG 协议也是一种南向接口。

1.2 版本的 OF-CONFIG 协议支持 OpenFlow1.3 版本的交换机的主要配置如下:

配置 datapath(在 OF-CONFIG 协议中称为 OpenFlow 逻辑交换机)连接的控制器信息,支持配置多个控制器信息,实现备份。配置交换机的端口和队列,实现资源的分配,远程改变端口的状态以及特性,完成 OpenFlow 交换机与 OpenFlow 控制器之间安全链接的证书配置,发现 OpenFlow 逻辑交换机的能力,配置 VXLAN、NV-GRE 等隧道协议。

OF-CONFIG 采用 XML 来描述其数据结构,并通过 NETCONF 协议来传输其内容。其顶层的数据结构图如图 5-13 所示。OF-CONFIG 的数据模型主要由类和类属性构成,其核心是由 OpenFlow 配置点对 OpenFlow 交换机的资源进行配置。在 OF-CONFIG v1.0 中,定义了 OpenFlow 端口和 OpenFlow 队列两类资源,它们隶属于各个 OpenFlow 交换机。每个 OpenFlow 交换机中包含多个逻辑交换机的实例,每个逻辑交换机可以对应一组控制器,同时也可以拥有相应的资源。另外,数据模型中还包含一些标识符,多数由 XML ID 标识。当前,这些 ID 都是一个字符串定义的唯一标识,以后有可能使用 URN(universal resource names,统一资源名称)作为标识。利用 XML 定义数据模型,能够具有较好的扩展性,同时也更便于利用软件进行实现。

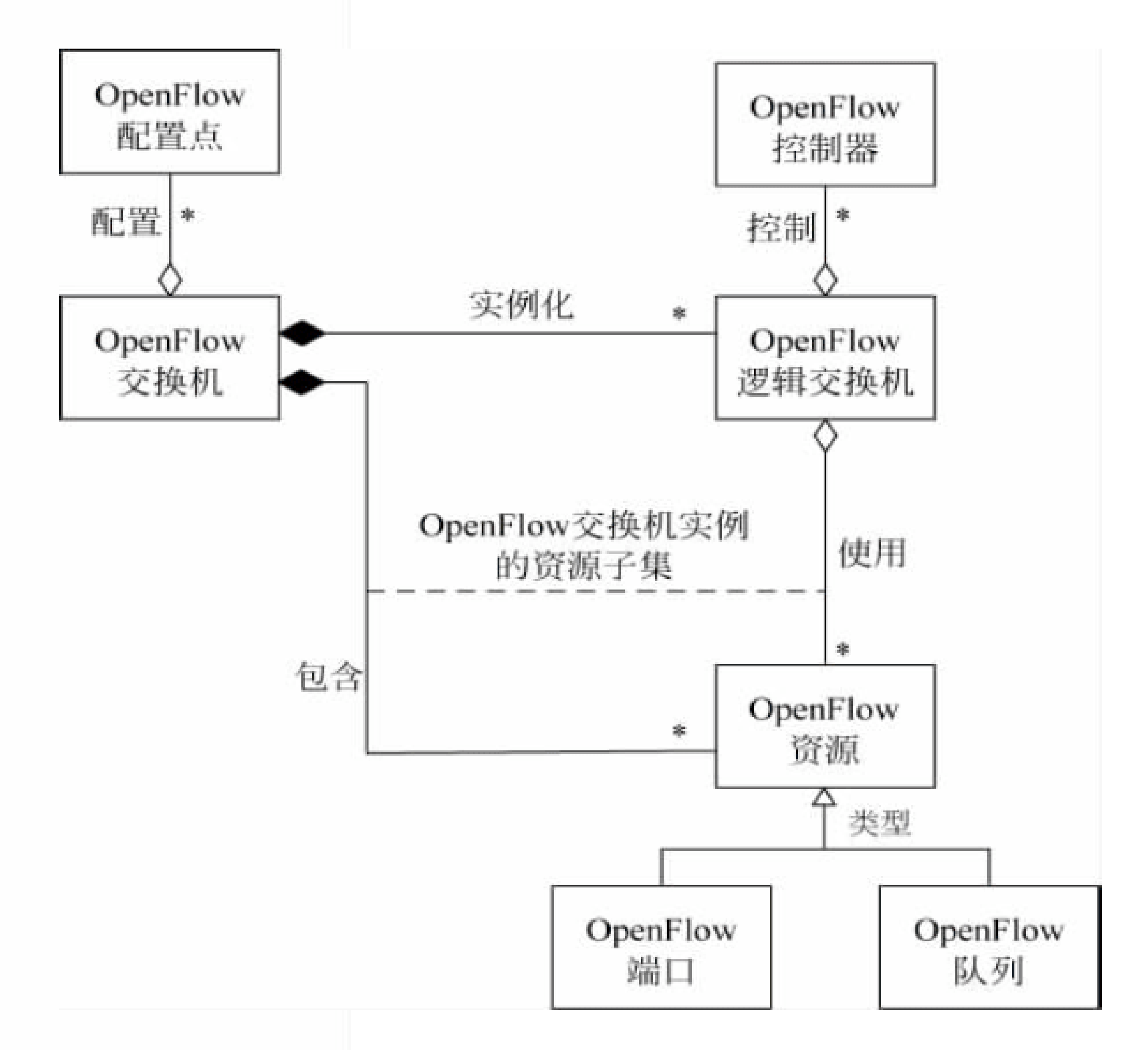


图 5-13　OF-CONFIG 顶层的数据结构图

作为 OpenFlow 的伴侣协议，OF-CONFIG 很好地弥补了 OpenFlow 协议规范之外的内容。在 OpenFlow 协议的 SDN 框架中，OF-CONFIG 需完成交换机的配置工作，包括将其连接到指定的控制器。当交换机和控制器连接建立之后，将通过 OpenFlow 协议来传递信息。从面向对象的角度看，OpenFlow 协议规范的范围仅负责指导交换机对数据流进行操作而无法对交换机的资源进行配置，而配置部分工作由独立的 OF-CONFIG 协议来完成，这个设计非常符合面向对象的设计理念。

作为伴侣协议，OF-CONFIG 协议是对 OpenFlow 协议的补充。其设计动机、设计目的和实现方式等都不一样。同时，OpenFlow 逻辑交换机的某些属性可以通过 OpenFlow 协议和 OF-CONFIG 协议两种方式来进行配置，所以两个协议也有相互重叠的地方。

OF-CONFIG 是 ONF 提出的 SDN 架构实现中的重要技术，与 OpenFlow 之间存在密切的关系。因此随着 OpenFlow 标准的演进，OF-CONFIG 的版本也与其保持同步。

5.2.4　SDN 硬件交换机

SDN 交换机位于数据层面，负责数据流转发，基本具有传统交换机的功能，通常可以采用硬件和软件两种方式进行转发。本节对硬件交换机进行介绍，交换机设备核心竞争力首先体现在交换芯片的性能上。常用的交换芯片技术有通用 CPU、专用集成电路(application-specific integrated circuit，ASIC)、现场可编程门阵列(field programmable gate array，FPGA)和 NP 等。一般对于硬件来说，交换机芯片的处理速度比 CPU 处理速度快两个数量级，比网络处理器(network processor，NP)快一个数量级，而且这种差异短时间内无法被克服[9]。但是在灵活性方面，芯片交换机远不如 CPU 和 NP 等可编程器

件。如何在既可以保证硬件转发速率又可以保证识别转发规则灵活性的条件下设计SDN交换机成为当前研究的热点问题。通用CPU功能易于扩展，理论上可以实现任何网络功能，但是数据处理性能不高。

下面对几种硬件方式实现的SDN交换机特点进行分析。

1. 采用ASIC芯片实现的交换机

ASIC芯片按需专门制定硬件电路，可高效率地实现某些网络功能，单颗芯片就可以实现几百兆PPS(packet per second)以上的处理能力，但ASIC芯片通用能力差，一旦开发完毕无法扩展到其他应用上，添加新功能需要芯片研发公司花费较长的开发周期，所以适用于实现网络中各种成熟的协议。SDN交换设备所需要实现的协议相对简单固定，但是需要较高的稳定性，因此采用ASIC芯片实现OpenFlow交换机是目前比较主流的商用化方式。然而，由于其研发代价，仅有少数巨头网络才倾向于自己研发专用ASIC芯片，其他大部分交换设备厂商使用专业芯片制造商生产的ASIC芯片。

如图5-14所示为一个典型的ASIC芯片处理示意图。网卡收到数据以后，经过集成在系统中的一些芯片直接处理而不经过主CPU，由这些芯片来完成数据流控制的功能，如缓存、流表项规则匹配、执行指令等，数据不经过主CPU处理也不使用中断机制。流表和转发规则全都固化在芯片中，只要ASIC设计得当，OpenFlow交换机可以获得很高的转发性能。

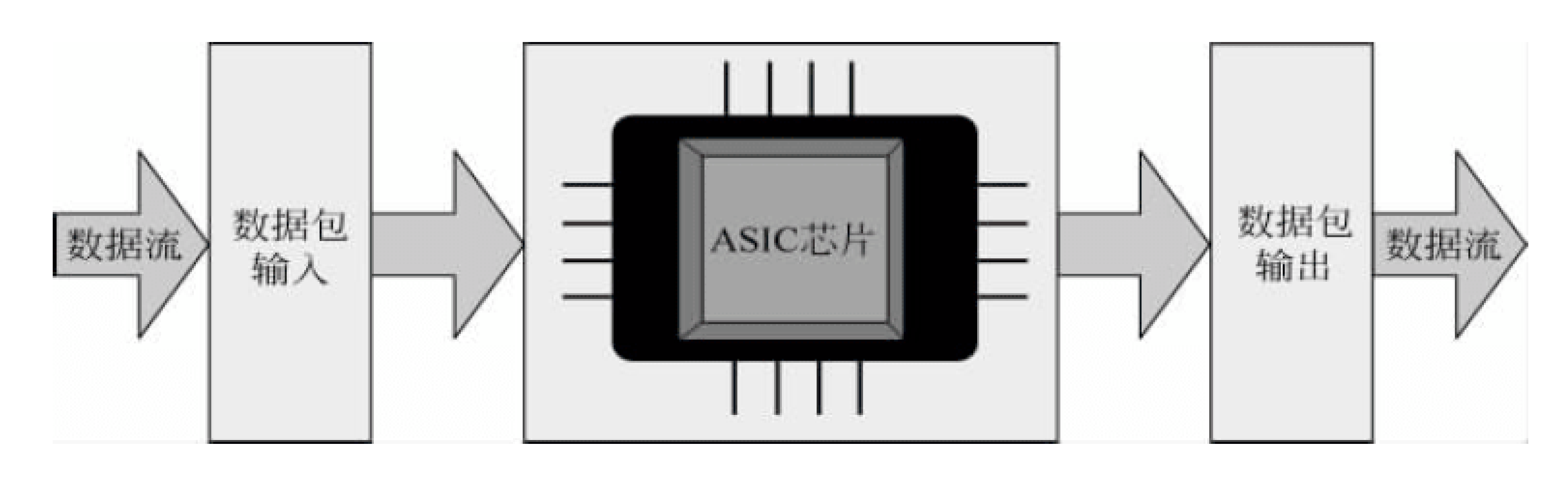


图5-14 ASIC芯片处理示意图[9]

推出基于ASIC芯片的品牌交换机的设备供应商主要有老牌网络供应商，如Cisco、Juniper、IBM等，这类公司具有自己的市场优势及技术基础，但是在推出SDN产品时，仍然沿袭传统商业模式，即将硬件与软件、应用、服务捆绑销售。典型代表产品有NEC IP8800系列交换机、IBM RackSwitch G8264交换机、HP SDN系列交换机、Arista 7150S系列和7500E交换机、DCN CS16800系列交换机、Cisco Nexus 9000系列交换机、Juniper EX9200系列交换机、H3C S12500系列交换机等。

实现网络可编程不得不提到白盒交换机，白盒交换机也叫做“裸机交换机”。它是与传统的品牌交换机相对应的。白盒交换机仅仅提供硬件，需要额外购买软件进行安装，它允许企业在不改变硬件的情况下自由更换NOS，提供了更多自由给企业网络运营。因此，白盒交换机可以认为是基于软硬件“分离”的思想实现的，而SDN促进了网络厂商在他们的交换机中增加网络编程功能，可以使用应用编程接口和其他基于标准的协议如OpenFlow，这意味着网络管理者不再需要命令行来配置交换机，而是使用类似于预定义的脚本或者第三方应用，减少了对某种CLI语句的依赖。这为白盒交换机的原始设备商

(original design manufacturer,ODM)提供了市场。可以期待未来的网络设备市场上,用户可以像挑选 PC 一样挑选各个厂商生产的网络设备,一旦网络设备软硬件接口标准化实现,ODM 厂商就可以购买芯片厂商生产的交换芯片,按照用户的要求来生产白盒交换机,用户根据自己的需求安装网络操作系统与应用程序,同时实现其与 SDN 控制器的交互,使得网络管理人员可以编程控制设备转发行为。目前盛科、Pica8、网锐均已推出基于 ASIC 芯片的 SDN 白盒交换机。

2. 采用 FPGA 实现的交换机

作为一种门阵列芯片,FPGA 支持可擦写,可以通过编程改变电路结构,以此实现不同的网络功能,但其不足之处是处理能力有限,难以实现大规模的网络转发,因此更适合于科研和验证。在 OpenFlow 领域,它们最多只能用作补充,以及用来验证技术可行性。斯坦福大学刚开始研究 OpenFlow 这个项目的时候,用的就是基于 FPGA 开发的 NetFPGA 可编程平台。

NetFPGA 是一种由美国斯坦福大学开发的低成本、可重用的硬件平台,其设计目标是开发出模块化程度高、开放性更强、可重构的网络处理器。最大限度地减少网络研究的任务量。如图 5-15 所示[9]为 NetFPGA 上 OpenFlow 交换机的实现过程。其中包括负责对数据包的包头进行解析并把解析的信息组装成流表项的对应格式的数据包预处理模块、负责主机与 OpenFlow IP 核通信的主机接口模块、用来访问(读或者写)交换机的流表并响应各个端口对其发起的所有请求的流表控制器模块、根据流表控制器发送的转发行为列表转发数据并更新其包头域与数据包长度的行为处理模块。

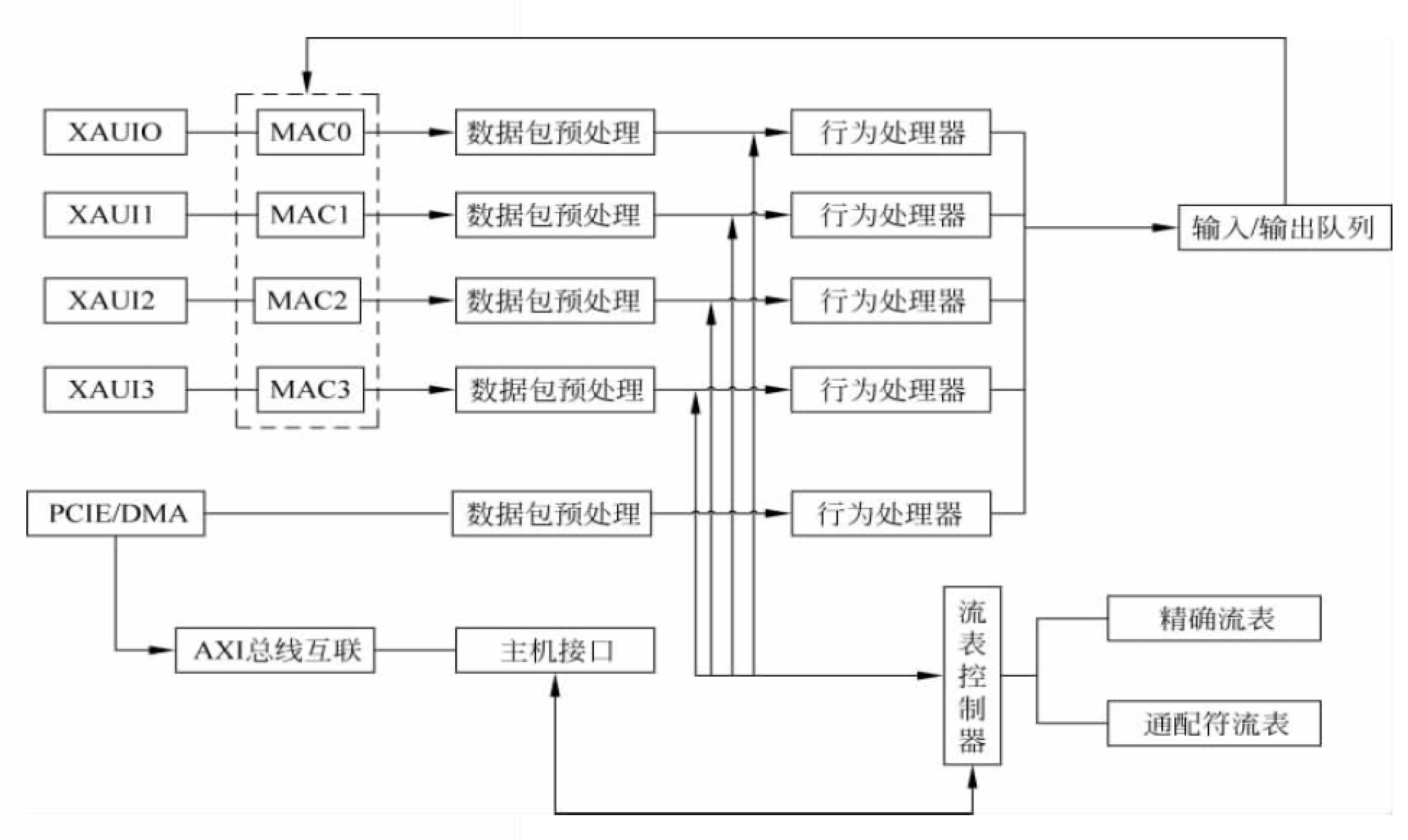


图 5-15 NetFPGA 上 OpenFlow 交换机的实现过程
(引自参考文献[11])

3. 采用 NP 实现的交换机

NP 作为一种可编程的处理器，利用众多并行运转的微码处理器进行复杂的多业务扩展，适用于实现各种创新或未成熟的业务，但是在网络厂商使用 NP 进行产品设计时需要投入大量的开发人员，同时性能和 ASIC 相比依然存在差距。华为作为国内最大的网络设备商推出了自己研发的以太网处理芯片——ENP 芯片，并于 2013 年 8 月发布了基于该芯片的敏捷网络架构以及全球首款敏捷交换机 S12700。2014 年 4 月华为和北京电信合作完成了全球首个运营商 SDN 商用部署，将 SDN 成功应用于 IDC 网络，该次 SDN 部署中使用的交换机就是华为 S12700 系列敏捷交换机。

5.2.5 SDN 软件交换机

软件交换机又叫做虚拟交换机。目前 OpenFlow 及其相关的协议正在不断演进改善中，硬件交换机研发周期长、灵活性低下成为主要问题，随着当前通用处理器性能的提升，基于软件的网络设备已经能够满足很多场景中的网络传输需求，因此 OpenFlow 软件交换机以低成本与灵活配置的优点成为 SDN 网络在科研实验平台以及中小型网络中部署的主要选择。下面主要介绍几款典型的基于软件的网络设备。

1. Open vSwitch(OVS)

OVS 是一个高质量、多层的虚拟交换机，遵循 Apache 2.0 许可证。目的在于让大规模网络自动化可以通过编程扩展，能够支持多种标准的管理接口和协议，如 NetFlow、sFlow、SPAN、RSPAN、CLI 等，还可恶意支持多个物理服务器的分布式环境，这一点和 VMware 的 vNetwork 分布式虚拟交换机或者 Cisco 的 Nexus 1000V 类似，但是 VMware 和思科的虚拟交换机是提供一个集中式的控制方式，而 OVS 是一个运行在每个实现了虚拟化的物理机器上的独立的 vSwitch，同时提供了远程管理。OVS 支持 OpenFlow 协议，可以通过流表来管理交换机的行为，并可以用来组成虚拟网络。

OVS 内部组件如图 5-16 所示。它主要包括三部分：OVS 的数据库 ovsdb-server，用来存储虚拟交换机的配置信息，它与管理者和 ovs-vswitchd 交换信息使用了 OVSDB (JSON-RPC)的方式；OVS 的核心部件 ovs-vswitchd，它和上层控制器通信遵从 OpenFlow 协议，它与 ovsdb-server 通信使用 OVSDB 协议，它和内核模块通过 netlink 通信，支持多个独立的 datapath(网桥)，通过更改 flow table 实现了绑定和 VLAN 等功能；OVS 的内核模块(ovs kernel module)，负责处理包交换和隧道，缓存 flow，如果在内核的缓存中找到转发规则则转发数据包，否则发送到用户空间去处理。

Open vSwitch 的运行原理为：内核模块实现了多个 datapath(类似于网桥)，每个都可以有多个 vports(类似于桥内的端口)。每个数据路径也通过关联一个流表(flow table)来设置操作，而这些流表中的流都是用户空间在报文头和元数据的基础上映射的关键信息，一般的操作都是将数据包转发到另一个 vport。当一个数据包到达一个 vport，内核模块所做的处理是提取其流的关键信息并在流表中查找这些关键信息。当有一个匹配的流时它执行对应的操作。如果没有匹配，它会将数据包送到用户空间的处理

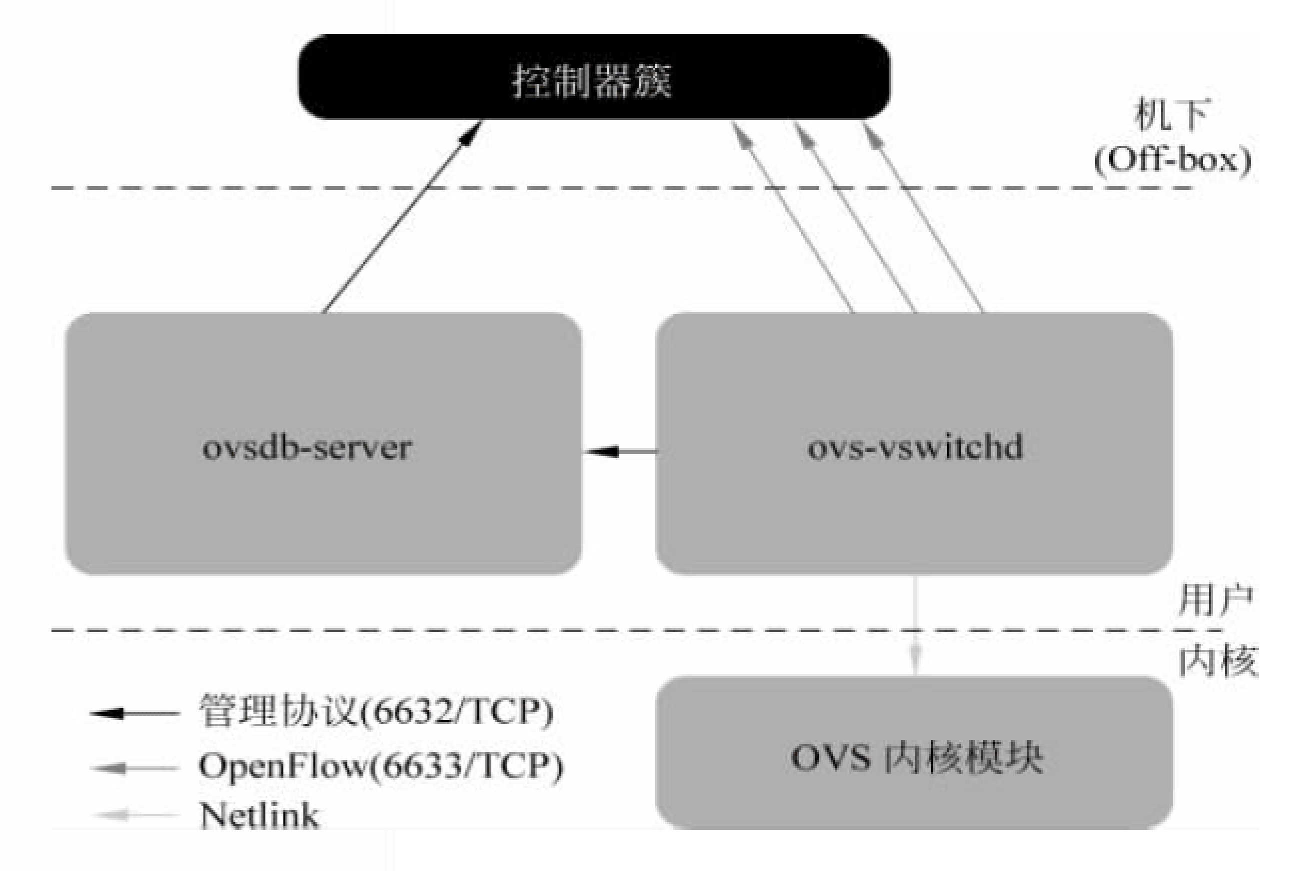


图 5-16　OVS 内部组件

队列中(作为处理的一部分,用户空间可能会设置一个流用于以后碰到相同类型的数据包可以在内核中执行操作)。Open vSwitch 实现的严密流量控制很大部分上是通过 OpenFlow 交换协议实现的。OpenFlow 使网络控制器软件能够通过网络访问一个交换机或路由器的数据路径。网络管理员可以使用这个技术在一台 PC 上远程控制数据管理,这样他们就能够进行精细的路由和交换控制,并实现复杂的网络策略。有了 Open vSwitch 中的这些远程管理功能,多租赁云计算的集成商和供应商就能够向客户提供在一台 PC 上持续管理各自虚拟网络、应用和策略的功能[12]。

2. Indigo

Indigo 是 Big Switch Network 根据斯坦福大学的 OpenFlow 参考方案,使用 C 语言实现的一个开源 OpenFlow 实现方案,运行于物理交换机之上,能够利用以太网交换机专用 ASIC 芯片的硬件特性,以线速运行 OpenFlow,支持多达 48 个高速率 10Gb/s 端口,并支持可扩展的网络虚拟化应用,它是使用 OpenFlow 控制器的跨越多服务器的分布式结构,类似 VMware 的 vNetwork、Cisco 的 Nexus、Open vSwitch。

3. Pantou(OpenWRT)

Pantou 基于所发布的 BackFire OpenWrt 软件版本(Linux 2.6.32),该实现方案可以把商用的无线路由器或无线接入点设备变为一个支持 OpenFlow 的交换机。它把 OpenFlow 作为 OpenWrt 上面的一个应用来实现。Pantou 的主要组件有 Linux 内核、uCibe 和 BusyBox,这些组件都经过优化以适合存储空间和内存都有限的家用路由器。目前,Pantou 支持的设备包括普通的 Broadcom 接入点设备、部分型号的 LinkSys 设备,以及采用 Broadcom 和 Atheros 芯片组的 TP-LINK 的接入点设备。

4. OpenFlowClick

Click 由美国 MIT 大学 Eddie Kohler 博士提出,MIT 计算机技术系并行与分布式操

作系统实验室开发完成了 clickl。他们深入研究了 Click 的扩展性与性能，并不断进行改进和发展[13]。第 4 章中已经提到，Click 是一个能灵活配置的软件路由结构，它通过一系列包处理模块(网络元素 Element)按照特定方式组合来实现网络功能，网络元素包括最简单单一的功能，如排队、调度、分类、复制等。OpenFlowClick 是在 Click 内部开发的一个 OpenFlow Element 组件，通过 OpenFlow 控制器动态、可选择的控制不同特性的网络流量在 Click 中的处理顺序，可以同时对数据分组和数据流进行处理的软件，即通过控制器中的逻辑控制对数据分组进行处理，通过将数据流与 OpenFlow 中的流表进行匹配对具有相同流特性的数据流进行处理，以提供给网络研究人员不同的控制方式。

5. LINC

LINC 是由 FlowForwarding 主导，基于 OpenFlow1.2 和 1.3.1 版本遵循 Apache 2 许可的一个交换机开源实现项目，架构采用流行的商用 x86 硬件，可运行于多种平台上，如 Linux、Solaris、Windows、MacOS，在 Erlang 运行环境的支持下，还可以运行于 FreeBSD 平台。

6. Of13softswitch

Of13softswitch 是一个与 OpenFlow1.3 版本规范兼容的用户空间的软件交换机实现方案。它是基于爱立信的 TrafficLab1.1 版软交换产品实现的与 OpenFlow1.3 版本规范兼容的用户空间的一个软件交换机方案。该软件最新版本的交换机包括交换机实现方案(ofdatapath)、用于连接交换机和控制器的安全信道(ofprotocol)、用于和 OpenFlow1.3 之间进行转换的库(oflib)以及一个配置工具(dpctl)。该项目由位于巴西的爱立信创新中心(ericsson innovation center)提供支持，并由同爱立信研究部门展开技术合作的 CPqD 提供维护[13]。

7. POFSwitch

POFSwitch 是由华为公司采用 BSD 许可基于 Linux 系统 C 语言实现的虚拟交换机，POFSwitch 与 POFController 协同工作，增强 OpenFlow 协议支持协议无感知转发[14]。

5.3 SDN 控制平面及相关接口技术

数据平面设备构成了网络的基础，相对传统网络而言，SDN 架构将数据平面交换设备的控制平面迁移到了集中化的控制器中，使得控制平面类似于网络中的“大脑”，所有的控制逻辑安置在网络应用与控制器上以构成控制平面。控制平面负责对底层网络资源的统一控制，同时向上层业务应用层提供开发的北向接口以实现网络控制的可编程性。之前提到，SDN 的基本概念包括可编程性、控制面与数据面分离以及开放接口，从这些概念上分析，SDN 的控制器是理想化的 SDN 框架的具体化，是 SDN 框架的一种具体反映。下面首先分析 SDN 控制器的架构，然后介绍控制器设计中的接口问题，最后根据理想的控制器架构，介绍目前几款广泛的商用或者开源控制器产品。

5.3.1 控制器架构

一般来说，控制器是一个软件系统或者系统集合，目的是实现对底层设备的功能抽象，提供给开发者开放的应用接口，以根据业务需求设计各种各样的网络应用。类比于计算机体系，可以认为 SDN 架构中控制器是网络的操作系统，以“操作系统”的形式运行在网络硬件设备资源上，有助于网络的自主化管理，提供了灵活控制网络的实现方式。而网络应用程序可以认为是运行在操作系统上的软件。和计算机系统一样，模块化的架构难以支撑复杂繁多的功能，导致网络功能的可扩展性变差，出于 SDN 社区发展协会和 OpenFlow 控制器目前的运营经验（Beacon）提出层次化的操作系统体系架构，即根据模块实现功能的差异，将其分类放置于不同层：基础模块放在底层，核心模块作为第二层，其余模块根据分类情况依次叠加。按照这一思路，首先对几款典型的控制器架构进行分析。目前几款主流的 SDN 控制器的架构和设计模块如表 5-5 所示。

表 5-5　几款主流控制器的模块功能分析

控制器 组件	OpenDaylight	OpenContrail	HP VAN SDN	Onix	Beacon
基础网络服务	拓扑/状态/交换管理，主机追踪，最短路径转发	路由，租户隔离	审计日志，警告，拓扑，发现	发现，多存储一致性，读状态，更新登记	拓扑，设备管理，路由
东西向 API	—	控制点	同步 API	分布式 IO 模块	尚未出现
集合插件	OpenStack Neutron	CloudStack，OpenStack	OpenStack	—	—
管理接口	GUI/CLI，REST API	GUI/CLI	REST API shell/GUI shell	—	Web
北向接口	REST，REST-CONF，Java APIs	REST-APIs（配置，操作和分析）	REST API，GUI shell	Onix API(通用目的)	API（基于 OpenFlow 事件）
服务抽象层	服务抽象层（SAL）	—	设备抽象层 API	网络信息基础（NIB）出入端口功能图表	—
南向接口或连接	OpenFlow，OVSDB，SNMP，PCEP，BGP，NETCONF	—	OpenFlow，L3 代理，L2 代理	OpenFlow，OVSDB	OpenFlow

可以看到，SDN 控制器应该提供以下功能[15]：

（1）网络状态的管理，并在某些情况下，管理和状态的分布可能会涉及数据库。这些数据库中存储着从网络控制组件和 SDN 相关应用软件获得的控制信息，包括网络状态、一些临时配置信息、拓扑学习和会话控制信息。在某些情况下，可能会通过多个控制器，来实现目标驱动（purpose-driven）的数据管理流程（如关系型和非关系数据库）。在其他情况下，也可以采用其他内存数据库（in-memory database）。

（2）高层的数据模型可以获取管理资源、策略和控制器提供的其他服务之间的关系。

很多数据模型都是使用 Yang 建模语言构建的。

(3) 控制器公开地向应用程序提供现代的、具有 RESTful(representational state transfer)性质的应用程序编程接口(API)。这有利于大多数控制器与应用程序间的交互。此接口来自于描述控制器的服务和功能的数据模型。在某些情况下,控制器和它的 API 作为开发环境的一部分以根据模型生成 API 代码。有些系统更进一步,提供了强大的开发环境,扩展其核心功能并为新模块推出其后续 API,包括支持控制器功能的动态扩展。

(4) 在控制器和网络组件的相关代理之间应使用安全的 TCP 控制会话。

(5) 基于标准的协议应当支持网络组件中的应用驱动的网络状态。

(6) 应具备设备、拓扑和服务的发现机制、路径计算制度,以及可以提供其他以网络为中心(network-centric)或以资源为中心(resource-centric)的信息服务。

同样从表 5-5 中可以看到,对于控制器,目前的研究主要集中在网络功能基元、服务以及数据与控制层面之间和控制平面与控制器之间的接口方面。表 5-5 中每一行代表一个组件,将其考虑为模块化和可扩展的控制平面的实现方式。在一个高度多样化的环境中,不同特性和组件被用于不同的控制平台上并不奇怪,在具有众多竞争对手的情况下,SDN 控制器都希望能做到发展前沿。另外需要注意的是,并非所有的组件都适用于所有的平台(例如,东西向接口 APIs 不需要集中化控制器,典型性代表如 Beacon)。事实上,许多平台都有具体的利润市场,如电信公司和云提供者,所以不同的商家需求也是不同的。

基于以上分析,尝试提取并提供 SDN 控制平面清晰系统的架构剖析图。如图 5-17 所示为典型的控制平面的组成。

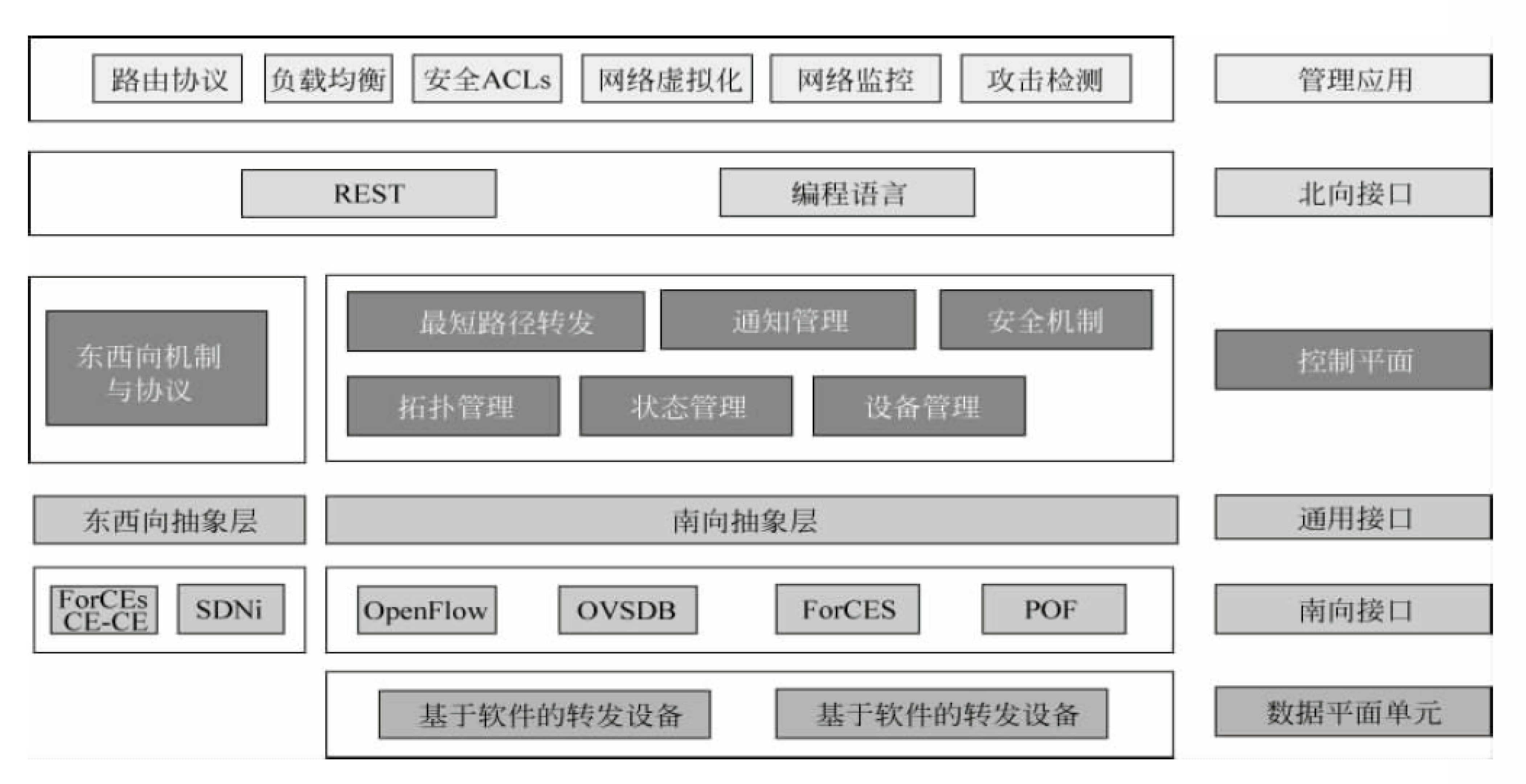


图 5-17　控制平面的组成

如图 5-17 所示,一个典型的控制平面存在至少三个相对定义良好的层:①应用、编排和服务;②核心控制功能;③南向通信单元。和上层的连接基本基于诸如 REST API

这样的北向接口以及诸如 FML、Frenetic 和 NetCore 这样的编程语言。在低层部分的控制平面包括南向接口 API 和协议插件接口这些基本转发单元。而控制平面的核心功能，可以被表征为其基础网络服务功能和多种接口的组合。同样以计算机操作系统来类比，控制器应该提供所有基本的网络服务功能，这些功能类似于操作系统的基础服务，如程序执行、I/O 操作控制、通信、防护等。这些服务由其他操作系统服务和用户级应用程序所使用，以类似的方式，像拓扑、统计、通知、设备管理结合最短路径转发与安全机制这样的必不可少的功能，可能会用于构建控制器逻辑。例如，通知管理者应该能够接收、处理和转发事件（这些事件包括警报通知、安全警报、状态更改）。安全机制与此不同，因为它是提供执行服务和应用程序之间的基本隔离和安全的关键组件。例如，由高优先级业务生成的规则不应该由具有较低优先级的应用程序创建的规则被覆盖。

5.3.2 控制器设计特性分析

显而易见，SDN 控制器的设计需要综合考虑多方面的需求。具体而言，其设计的核心要素如图 5-18 所示。

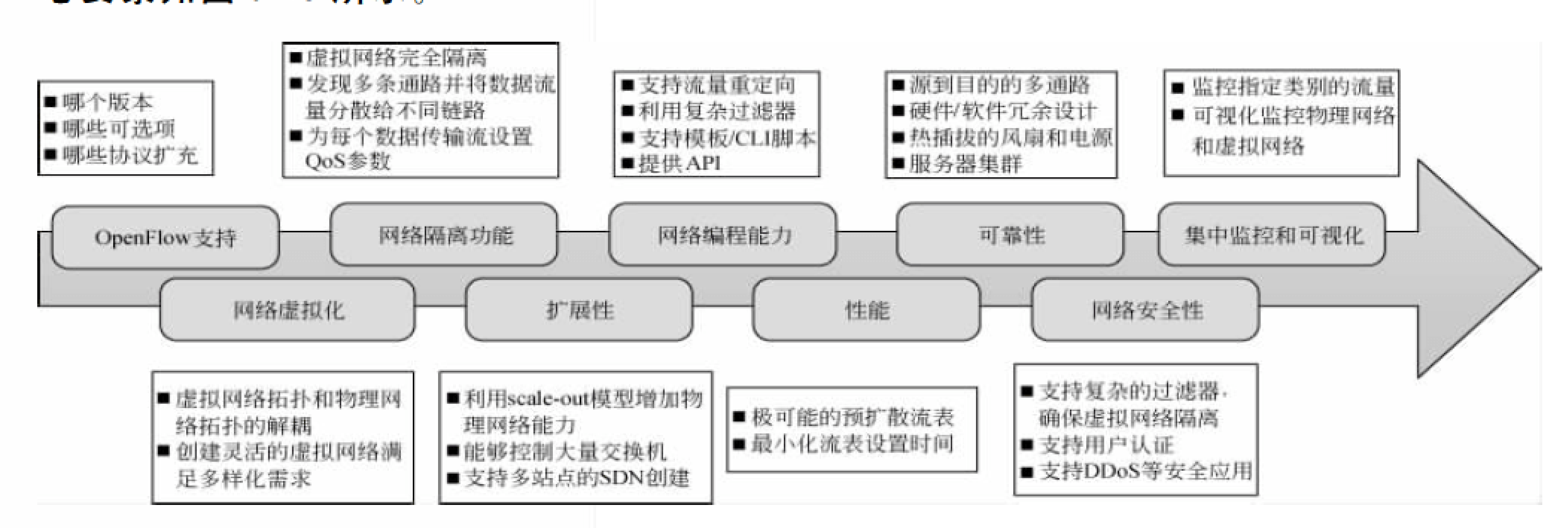


图 5-18 控制器的核心设计要素

在这些 SDN 控制器的核心能力中，有些是传统网络设备已经关注的，如扩展性、性能、安全性、集中监控及可视化、可靠性等。同时，也有一些是专门面向 SDN 网络的，如 OpenFlow 支持、网络虚拟化和网络编程能力等。

通常，SDN 控制器的功能是将网络环境中的控制与数据平面互相分离。换句话说，控制器将告诉网络设备如何转发流量（控制平面），但是它们并不真正转发这些流量（数据平面）。这种情况在 OpenFlow（OF）网络中非常常见。在 OpenFlow 网络中，SDN 控制器主要用于对网络设备中的 OF 表单进行编程，OF 交换机接收数据包并根据流表处理这些数据包。但是如果流表中的数据包没有匹配的条目，将会怎样？在这种情况下，OF 交换机将把数据包发给 OF 控制器，这实际上相当于在问“我应该怎么处理这些数据包？”。OF 控制器来决定当数据包与流匹配时交换机应当做些什么，并对交换机进行编程。这一程序被称为“流安装”。由于扩展方面的需求，SDN 控制器每秒的流安装量受到了高度重视。通常，流安装在 SDN 中会存在一个性能瓶颈。但是不要想当然地认为，在大量交换机和希望控制的庞大微流数量共同作用下，它们很快就会超过控制器流安装能

力。必须牢记,并不是每个流都需要与控制器联系。只有那些还没有被识别或编程的流才需要这一步骤,而这通常只是例外情况。

网络虚拟化是SDN的一个重要价值,对于SDN面临的用户需求而言,网络虚拟化需要是端到端(end-to-end),并且可以像服务器虚拟化技术一样抽象和池化网络资源,这些能力能够创建出满足租户需求的、拓扑与底层物理网络解耦的虚拟网络,同时还使得管理人员能够动态地创建基于策略的虚拟网络来满足广泛的需求,将虚拟网络和物理网络解耦的另一个优势是网络管理人员可以在不影响现有数据流的前提下对物理网络进行调整。另一方面是网络虚拟化带来的网络隔离,SDN控制器必须能够通过集中控制与自动配置实现虚拟网络的完全隔离。

关于网络可编程性,API是从控制器中获取和向控制器发送信息的机制,网络应用程序会通过北向API告诉控制器它们需要什么。API是自动化和编配的关键。例如,你的机构能够以多高的效率利用控制器API? 对于正在寻求整体解决方案的用户,他们可能并不关心这个。但是对于那些希望编写自己的网络应用程序的用户来说,这一点非常关键。

目前还没有创建SDN控制器的统一方式,也没有一套关于SDN必须具备哪些功能的要求。接触的控制器和架构越多,就越会发现它们之间存在着相当大的差异。这些都是意料之中的事情。由于SDN还是一项新兴技术,因此厂商都在发布能够代表其SDN观念的控制器,以尽可能地在市场中突出自己,从而力争成为市场领导者。

当前商业化的控制器的供应商有VMware (vCloud/vSphere)、Nicira (NV)、NEC (Trema)、Big Switch Networks(Floodlight/BNC)以及Juniper(Contrail)。当然还有几种开源控制器(某些厂商同时提供开源的和商业的控制产品)。除了使用OpenFlow协议和专有协议,还有些SDN控制器利用IP/MPLS网络的功能来创建MPLS VPN,作为数据中心或广域网(WAN)上MPLS LSP Overlay的三层到三层(layer 3-over-layer 3)的租户分离模型(tenant separation model)。不能忽视这样的观点:网络管理解决方案中,基于NETCONF的控制器几乎没有区别,或者说基于Radius/Diameter的控制器(如PCRF和/或TDF),在移动环境中也充当SDN控制器的角色。在这样的事实下,南向协议变得更加独立,并能够创建临时网络/配置状态。

数据中心业务流程中原有SDN应用催生了SDN控制器,并使其作为综合解决方案的一部分。由于这样的原因,SDN控制器侧重于数据中心的资源,如计算、存储和虚拟机映像,以及网络状态的管理。最近,出现了专门从事网络抽象的管理,并能够对数据中心进行资源管理的SDN控制器,它们是通过开源API(如OpenStack、CloudStack)的支持来实现这些功能的。SDN应用在数据中心之外的扩展,以及某些网络领域内并没有在管理方案中将计算和存储这类虚拟资源紧密结合,催生了这类控制器。

虚拟交换机或路由器代表网络环境中的最低共同标准,并且一般其转发表项的数量比硬件实现的同等设备要少。虽然在技术上能够在VM(virtual mechine,虚拟机)服务中支持大型表,但它们真正受限制的地方并不是VM服务。尤其是因为,集成表的规模和虚拟层的管理功能通常只能在特定用途的路由器或交换机中以硬件方式实现。在传统特定用途的网络组件中,更为简化的基于hypervisor的转发结构中没有RIB/FIB组合存在的空间。这种情况下的分布式控制范式需将分布式网络信息归结为这几个条

目——这些信息来自用户空间代理(user-space agent)(终端主机结构的一部分,建立一个进程并将其作为一个VM服务运行)或SDN控制器。在后一种情况下,SDN控制器的作用可能是在分布式环境中的一个代理,或是集中式环境中管理决策下的流配置代理(flow provisioning agent)。采用这种方式,控制器才能在网络的管理层上,通过一个常规的网络OSS(open source software,开源软件)公开。

对于在数据中心主机上的软件交换机/路由器,SDN控制器是一个关键的管理界面。SDN控制器提供一些管理服务(以及配置和发现功能),因为它们要负责临时的网络实体相关状态(通过代理),如分析和事件通知。从这方面来讲,SDN控制器将改变传统网络元素管理的定义。

5.3.3 控制器接口技术

SDN定义的要素之一——开放的可编程接口包括北向接口、南向接口与东西向接口三种接口类别。前面已经讨论了南向接口的概念及其协议规范,SDN通过南向接口实现数据网络中控制平面与数据平面的分离,以ONF的OpenFlow协议和IETF的ForCES、OVSDB、NETCONF+YOUNG等协议为典型代表;北向接口是用户业务以及各种网络业务开发者有效控制和利用网络资源的门户,同时也是通过控制器向上层业务应用开放的接口;东西向接口负责解决多个设备控制平面之间的协同工作的问题。下面介绍北向与东西向的接口技术。

1. 北向接口技术

SDN向上提供网络资源抽象,实现软件可编程控制的网络架构,上层的网络资源管理系统或者网络应用可以通过控制器的北向接口全局调度整个网络的资源状态并进行统一调度。因为北向接口是直接为业务应用服务的,因此其设计需要密切联系业务应用需求,具有多样化的特征。同时,北向接口的设计是否合理、便捷,以便能被业务应用广泛调用,会直接影响到SDN控制器厂商的市场前景。

与南向接口方面已有OpenFlow等国际标准不同,北向接口方面还缺少业界公认的标准,因此,北向接口的协议制定成为当前SDN领域竞争的焦点,不同的参与者或者从用户角度出发,或者从运营角度出发,或者从产品能力角度出发提出了很多方案。据悉,目前至少有20种控制器,每种控制器会对外提供北向接口用于上层应用开发和资源编排。虽然北向接口标准当前还很难达成共识,但是充分的开放性、便捷性、灵活性将是衡量接口优劣的重要标准。例如,REST API就是上层业务应用的开发者比较喜欢的接口形式。部分传统的网络设备厂商在其现有设备上提供了编程接口供业务应用直接调用,也可被视作是北向接口之一,其目的是在不改变其现有设备架构的条件下提升配置管理灵活性,应对开放协议的竞争。

1)开源控制器的北向接口解决方案

目前OpenDaylight、Floodlight、Ryu等开源实现的控制器均有自己的北向接口解决方案。

OpenDaylight采用REST API作为自身的北向接口,基于YANG构建模型,然后由

YANG Tools 开源工具生成对外的 API 以及部分 plug-in 框架代码。REST 从资源的角度来观察整个网络，分布在各处的资源由 URI 确定，而客户端的应用通过 URI 来获取资源的表示方式。获得这些表示方式致使这些应用程序转变了其状态。随着不断获取资源的表示方式，客户端应用不断地在转变着其状态。所谓表述性状态转移（representational state transfer）。YANG 是 2000 年由 Roy Fielding 博士提出的软件架构风格，初始作为 NETCONF 的建模语言被创造出来，作为一种针对网络应用设计的开发模式，可以降低开发复杂度，提高系统灵活性。在 YANG 模型中，网络中所有资源有统一且唯一的资源标识符（URI），所有资源通过超链接连接在一起，使用统一接口操作网络资源，网络资源请求结果提供多种标准表述格式（可扩展标记语言 XML）或者 JavaScript 对象表示法（JSON），同时服务器不保持客户端状态。

OpenDaylight 北向接口开发分为两类：可以复用的 API 驱动 SAL（AD-SAL）和模型驱动 SAL（MD-SAL）。AD-SAL 南北向接口均需要定义相应的 AD-SAL，因此影响模块可扩展性，因此 MD-SAL 上的北向接口为主要发展方向。OpenDaylight 北向接口 MD-SAL 开发流程首先需要用 YANG 定义一个 model，定义接口输入/输出参数，使用 Maven 编译时调用 YANG Tools 自动生成对应的 API，再生成 API 开放服务网关协议（OSGI）Bundle；然后实现接口代码，编写插件源代码，通过 Maven 编译生成 plug-in OSGI Bundle，最后部署到 OSGI 上，在全局编译时还需在对应的 pom.xml 文件中对它进行描述，使得编译时将对应的 Bundle 编译并生成对应的 jar，从而在控制器中添加功能，将对应的 jar 包放至控制器对应目录，控制器运行时接口会一起运行[16]。

Floodlight 北向接口也是基于 REST API，不仅提供了基本的网络状态查询、统计、配置接口，还提供了允许用户手动插入流的静态流推送器 API、创建虚拟网络的 API 以及防火墙 API。

Ryu 采用组件化的结构，由 Rython 语言编写，并提供大量库函数（组件）供 SDN 应用的开发。Ryu 针对北向的用户的应用，Ryu 框架也提供了 RESTful 管理的 API、REST API 以及用户通过 REST 或者 RPC 自定义的 API。其中，RESTful 管理 API 是用户通过相应组件配置管理 SDN 网络的接口，如可以通过 OF REST 组件 API 配置 OpenFlow 交换机；通过 Firewall 组件 API 配置防火墙等；REST API 是与 OpenStack 云编排系统对接的接口；同时用户也可通过自定义的方式实现 REST API 或 RPC API。

2）北向接口标准化进程

北向接口没有统一标准给上层应用带来了很大的困扰，造成学习成本的提升，标准化北向 API 将允许客户在一致的基础上视需求自行开发控制器上层应用与网络服务，加速 SDN 的大规模商用化。因此 ONF、IRTF 和 IETF 等相关工作组正在致力于推进北向接口的标准化。

2012 年，ONF 创建了一个关于北向 API 的讨论组，其目标是建立一个交付成果与时间线的正式工作小组。这个讨论组被并入 ONF 的架构与框架工作小组（architecture and framework working group）。目前 NBI 小组尚未发布相关的标准草案，但是已经制定了一个章程，其中包含一些激发北向 API 需求的用例、一个包括寻找北向 API 实例需求的纲要研究以及相关建议。并且在目标文档中指出，对于 SDN 北向接口应该给出不同层次的抽象以及接口范围。ONF 北向接口中，不同抽象层及接口范围需要发展不同

的 API，如应用于虚拟化管理及 QoS 等需要不同的 API。目前，针对北向接口中 SFC、VPN 等技术方案相关的标准还在进一步讨论中，同时基于意图(Intent)的北向接口的信息模型、数据模型、应用实例等也都在积极的进行中。

IETF 是全球互联网最具权威的技术标准化组织，主要任务是负责互联网相关技术规范的研发和制定，重点是定义设备间的协议及技术规范。随着 SDN 技术理念被业界广泛接受，IETF 也成立相应的工作组全面开展标准化工作，其研究更注重重用现有的协议和架构，立足于以演进的方式实现集中控制和网络的可编程。在北向接口标准的制定方面，IETF 也取得了很多进展。IETF 前期侧重于制定技术方案接口类北向接口，早期 IETF 成立 ALTO(application-layer traffic optimization)工作组，主要通过为应用层提供更多的网络信息，完成应用层的流量优化。已经发布 6 篇 RFC，包括需求、协议等内容，正在定义信息模型、数据模型等。

在 2013 年，成立 SFC 工作组，主要职责是定义如何将多个业务功能通过业务链串接起来形成不同的业务功能集合，满足不同场景的需求和部署场景。目前，已经发布 SFC 架构 RFC，并在努力制定相应的数据模型。2015 年成立 OPS 域的网络级技术接口的工作组 L3SM，旨在制定统一 L3VPN 部署的数据模型，以 L3VPN 为目标，探索 IETF 定义网络级业务统一接口的可行性。目前，已有多篇草案(drafts)阐述 VPN 部署的自动化管理以及统一数据模型。随着 Intent 概念被不断热议，IETF 也开始成立专门的工作组来制定 Intent 接口标准[17]。

2. 东西向接口技术

控制器之间的接口被称为东西向接口。随着软件定义网络的普及，单一控制器已经不足以满足大规模软件定义网络架构的管理需求。目前在保证控制全网信息状态的条件下，根据不同网络特性与服务需求提出的控制层架构主要分为分布式与集中式扩展结构：一是着重考虑网络稳健性，将整体网络中的多个控制器呈对等架构关系物理部署在各个网域中，所有控制器逻辑上均掌握全网状态，网络环境变化时进行同步更新；二是着重考虑资源利用率，在控制层部署局部控制器与全局控制器，局部控制器仅负责本网域中的网络状态，全局控制器通过与局部控制器的信息交互实现全网信息的管理和维护。

在该方面国内外已开展多种研究以满足大规模网络管理需要。2012 年，Dan Levin 等提出通过扩展控制器的数量可以解决单一集中控制的单点失效问题，也就是将控制器物理分布在整个网络中，仅需保持逻辑中心控制特性即可。2013 年，Stefan Schmid 等提出分布式控制器一般可采用扁平控制方式和层次控制方式进行扩展。对于扁平控制方式，所有控制器被放置在不相交的区域里，分别管理各自的网络。对于层次控制方式，局部控制器负责各自的网络，全局控制器负责局部控制器，控制器之间的交互可通过全局控制器来完成。Google 的分布式控制器 Onix 在所有节点之间维护了全局网络视图，实现分布式控制。此外，还定义了一套 API，用于定义具体的同步操作。面对不同的场景，如不同域之间的通信，可制定具体的同步数据细节来保障网络的安全和隐私。Onix 支持分区(partition)和聚合(aggregation)两种形式的网络拓展。HyperFlow 允许网络运营商部署任意多个控制器，并将这些控制器分布在网络的各个角落。控制器之间保持着物理分离而逻辑集中的特点，因此仍然保持 SDN 集中控制的特点。Kandoo 实现了层次分布

式结构。当交换机转发报文时，首先询问较近的局部控制器。若该报文属于局部信息，局部控制器迅速做出回应。若局部控制器无法处理该报文，它将询问全局控制器，并将获取的信息下发给交换机。该方式避免了全局控制器的频繁交互，有效降低了流量负载。但是在扁平控制方式中，虽然每个控制器掌握全网状态，但只控制局部网络，造成了一定资源的浪费，增加了网络更新时控制器的整体负载。

由于控制器架构可扩展趋势，实现控制信息交互的东西向接口逐渐成为 SDN 控制层协议的重要研究点之一。在学术领域，2013 年清华提出了 West-East Bridge 的概念。在相关论文中介绍了他们设计的东西向接口的功能，即基于高性能的一种对等网络间的网络信息交流机制，并完成了部署和测试。该架构允许控制器通过标准化插件的形式对其进行集成，从而实现异构 SDN 控制器间标准化的东西向通信。在 West-East Bridge 的设计中，控制器以“全网状”的拓扑形式进行组网，它主要实现控制器发现、网域视图信息维护和多域网络视图信息交互等功能。当网络视图发生变化时，该事件将会被发布到所有订阅其数据的节点。为保证数据的一致性，其节点之间为全连接关系。此外，West-East Bridge 还设计了虚拟的网络视图，可以满足某些 SDN 域对于安全和隐私的需求。

在东西向接口的商业化方面，Juniper 推出的 SDN 解决方案 OpenContrail 架构中，以内部边界网关协议(interior border gateway protocol，IBGP)作为东西向接口协议，控制节点之间通过 IBGP 交换路由，保证所有控制节点中控制信息的同步。华为提出一种用于处理 SDN 域之间通信的协议 SDNi。在其提交的草案中，定义了 SDN 域的概念和 SDNi 如何帮助域之间通信。目前 SDNi 已经在开源控制器 OpenDaylight 上作为应用实现。SDNi 需要在控制器之间交互 Reachability、Flow setup/tear-down/update 请求和包括带宽、QoS 和延迟等性能信息。SDNi 的数据交换可以基于 SIP 或者 BGP 协议实现，如 OpenDaylight 中就是基于 BGP 协议实现的。基于 SDNi 可以实现异构控制器协同工作，实现大规模网络的管理，实现跨域流量优化等应用。同时，中国移动发布的关于 SPTN 的白皮书中也有涉及多控制器协同工作的内容，然而目前仅仅处于白皮书阶段，并没有实际部署和测试。

以前控制平面多以单例控制器为主，由于控制器容量限制，缺乏控制平面的可扩展性，SDN 在大规模部署中将举步维艰，使得 SDN 控制平面能力欠缺成为 SDN 网络规模受限的最大原因之一。后来 OpenDaylight、ONOS 等控制器的出现，使用了集群技术，使得 SDN 控制平面性能得到了提升，是当下解决 SDN 控制平面性能不足的主要解决方案之一。但是这些解决方案只能在同一控制器之间实现性能扩展，异构控制器之间协同工作依然是显著问题。东西向接口作为控制平面可扩展性的核心技术，目前研究尚处于探索阶段，缺乏行业标准，理想化的东西向接口应该与控制器解耦以实现不同厂家控制器之间的通信，同时适应不同的控制层面架构，并兼顾考虑 SDN 域之间的安全隐私问题，因此，如何完成跨域之间的流量处理，实现多异构控制器之间的协同工作是未来研究的方向之一。

5.3.4 开源与商用控制器项目

目前工业界和学术界针对不同的网络环境已设计推出多款 SDN 控制器解决方案，

其中开源社区在 SDN 推动过程中贡献了重要力量，同时为了更好地推动 SDN 技术发展，针对具体的应用场景、具体交换机，一些 IT 企业推出了相应的商用服务器，这些控制器与开源控制器相比，拥有企业更加专业的支持与维护，具有更好的稳定性、可靠性以及性能。本节对 IT168 收集[18]的可追寻到的开源 SDN 控制器和商用 SDN 控制器，进行介绍。

1．开源控制器

目前业界主要的开源控制器如表 5-6 所示。

表 5-6　开源控制器调研分析

开源组织	控　制　器	编程语言	特 征 简 介
NOXRepo	NOX	C++/Python	一款原始的 OpenFlow 控制器，它有利于在 Linux 上进行快速的 C 控制器的开发
	POX	Python	支持 Windows、Mac OS 和 Linux 系统的 Python 开发，主要用于研究和教育领域
ON. Lab	SDN Open Network Operating System (ONOS)		由 SDN 发明者及斯坦福大学和伯克利大学领导者们创建的一个非盈利组织。该组织设立的 ONOS 项目旨在研发开源 SDN 工具
OpenContrail	OpenContrail Controller		由 Juniper 赞助的 OpenContrail 开源架构包括一个逻辑上集中但物理形态上分布的 SDN 控制器、虚拟路由、分析引擎，并且还发布了北向 API (Juniper 还支持 OpenDaylight)
OpenDaylight Project	最新版产品为 Helium(最初版产品为 Hydrogen)		该组织是一个 Linux 基金合作项目，OpenDaylight 正为增强网络可编程性研发一个开放的平台，希望能实现任何规模的 SDN 和 NFV 网络
	Open Mul		旨在提升性能、可靠性、灵活性以及易学性，它是一个 OpenFlow SDN 控制器平台，内核是一个基于 C 语言的多线程基础架构，用于托管应用的多层级北向接口
Project Floodlight	Floodlight Open SDN Controller		由 Big Switch Networks 发布的 Floodlight 是一个使用 OpenFlow 协议和 Apache 许可证的 SDN 控制器
	Ryu		一个软件定义网络架构，带有一个定义好的 API，该 API 旨在帮助程序员创建新的网络管理和控制应用。它支持标准的协议，包括 OpenFlow、NETCONF 和 OF-CONFIG
斯坦福大学	Beacon		一款跨平台、模块化、基于 Java 的 OpenFlow 控制器，该控制器支持基于事件和线程的操作
	Trema		Trema 是用 Ruby 和 C 来开发 OpenFlow 控制器的架构

2. 商用控制器

商用控制器分析汇总如表 5-7 所示。

表 5-7　商用控制器分析汇总

公　司　名	控　制　器	特 性 简 介
Active Broadband Networks	Active Resource Controller	旨对 NFV 基础设施提供实时控制，为服务和应用可视性提供 IP 流遥测数据，为个性化服务管理、动态服务精简和控制机制提供大数据技术，以便随时根据客户的授权更改网络状况和服务使用情况。ARC 是该公司 Software-Defined Broadband Network Gateway 的组件
Adara Networks	Sky	一款基于 OpenFlow 的 SDN 控制器。还有 Horizon，一款为 SDN 管理设计的元控制器，可用于多厂商多协议（无论虚拟还是物理）的网络
Big Switch Networks	基于 Project Floodlight 开源标准，遵守 OpenFlow 协议的 SDN 控制器	Big Switch Networks 要把这个控制器放入 Open SDN Suite 套件中，供数据中心运营商使用
Brocade Communications Systems	Vyatta Controller	一款基于 OpenDaylight 标准的开源控制器，以 OpenDaylight 代码研发，博科 Vyatta 控制器旨在为网络运营商带来可编程网络的灵活性，且为多厂商和虚拟机提供一个普通平台
Calient Technologies	Optical Topology Management Controller	用 OpenDaylight 代码编写，大量数据从数据包转向光纤交换机时，数据中心运营商可利用此控制器进行重新配置
Ciena	Agility Multilayer WAN Controller	一款基于 OpenDaylight 标准的控制器。控制器可让运营商优化广域网，以满足企业和云用户预料之外的带宽需求
Cisco Systems	Application Policy Infrastructure Controller (APIC)	该控制器可以对思科的 Application Centric Infrastructure SDN 架构进行自动化操作和管理
Cyan Inc	Blue Planet SDN 控制器	Blue Planet SDN 平台自 2012 年 12 月推出以来，已经部署到全球 154 个网络，其中不乏 Colt、KVH 和 NTT Americans 等客户。Blue Planet 将 WAN SDN 控制器用于多层级和多厂商自动化、路径计算、虚拟化、预配置、管理和控制功能。它与 Blue Planet 的 NFV 以及虚拟资源编排功能一起推动新的虚拟服务
CloudGenix	Software-Defined Enterprise WAN (SDE-wan)（目前仅提供 beta 测试版）	CloudGenix 是一家研发控制器的新创公司，该公司希望把企业 SDN 扩展到 WAN

续表

公　司　名	控　制　器	特 性 简 介
ConteXtream	ContexNet controller	用 LISP、OpenFlow、OpenDaylight、NV03、OpenStack 等提供基于标准的可扩展性。ConteXtream 已经研发出两个独立应用，分别是 ContexMap 和 ContextControl，合二为一即为一个 SDN 控制器
Coriant	Transcend SDN Solution	包括 Transport Controller、Packet Controller 和 SDN Network Orchestrator
CPlane Networks	CPlane Networks Controller	CPlane Networks 已经为 Openstack 云基础设施的部署研发了一个新的控制器
戴尔	Active Fabric Controller	戴尔的这款软件适用于企业级 OpenStack 部署，而且使用 OpenFlow 协议与数据中心交换机对接
Extreme Networks	Extreme OneController	基于 OpenDaylight 标准，Extreme Networks 希望借此帮助用户从原有网络迁移到 SDN。特别是与 USIgnite 联手推出了 Extreme SDN Innovation Challenge
惠普	惠普 Virtual Application Networks SDN Controlle	该控制器是给 OpenFlow 网络的一个统一控制器，但是也支持其他开放的可编程接口
华为	Smart Network Controller（也被称为 Smart OpenFlow Controller）	华为控制器适用于华为自己的 Netmatrix SDN 编排系统，且支持 OpenFlow、PCE、Netconf 和 BGP
IBM	IBM SDN for Virtual Environments	IBM 的控制器支持基于 OpenFlow 的物理网络
Inocybe Technologies	Infrastructure Controller	Inocybe Technologies 的控制器将分别基于 OpenDaylight 和 Openstack 协议的 SDN 和云控制器结合在了一个平台
Juniper 瞻博网络	NorthStar 和 OpenContrail	Juniper 有两个 SDN 控制器：自产的 NorthStar 和 Contrail SDN Controller，二者源自对 Contrail Systems 公司的收购。Contrail 控制器组件也可通过 OpenContrail 的开源选项获得
Metaswitch Networks	Gulfstream SDN Controller	Metaswitch Networks 的 Gulfstream 控制器适用于网络访问和大型数据中心架构。它支持的标准包括 OpenFlow1.3、REST、NETCONF 和熟悉的 CLI 接口。结合了开源和 Metaswitch 专属代码
NEC	ProgrammableFlow SDN Controller	NEC 是首个发布基于 OpenFlow 协议商用机控制器的公司。[现在它也有了一个控制协调程序，就是 Unified Network Coordinator。]该工具可帮助 OpenFlow 在数据中心内外的扩展。Unified Network Controller 管理着多个 OpenFlow 控制器，这些控制器又反过来管理交换机
Nuage	Virtual Services Controller	为 Nuage's Virtualized Services Platform 平台提供控制面板

续表

公　司　名	控　制　器	特 性 简 介
Pica8	支持 RYU OpenFlow 的开源控制器	日本巨头 NTT 集团也支持 RYU OpenFlow 开源控制器，而且其实验室还为其提供研发力量。Pica8 把这款开源控制器作为自己 SDN Starter 工具包的一部分提供给用户
Plexxi	Plexxi Control	Plexxi 的数据中心控制器可根据工作负载的需求动态优化网络
VMware	NSX Controller	NSX Controller 搭建于 VMware 的 NSX SDN 平台上，不能作为单独的产品供用户使用

参考文献

[1] 程莹，张云勇. SDN 应用及北向接口技术研究[J]. 信息通信技术，2014(1)：36-39.

[2] Specification，OpenFlow Switch. Version 1.0.0(Wire Protocol 0x01)[S]. 2009.

[3] Jain R. Internet 3.0：Ten problems with current Internet architecture and solutions for the next generation. In：Proc. of the IEEE MILCOM. 2006：1-9. [doi：10.1109/MILCOM.2006.301995]

[4] Tennenhouse DL，Wetherall DJ. Towards an active network architecture. In：Proc. of the IEEE DARPA Active Networks Conf. and Exposition. 2002：2-15. [doi：10.1109/DANCE.2002.1003480]

[5] Greenberg A，Hjalmtysson G，Maltz DA，Myers A，Rexford J，Xie G，Yan H，Zhan JB，Zhang H. A clean slate 4D approach to network control and management. ACM SIGCOMM CCR，2005，35(5)：41-54. [doi：10.1145/1096536.1096541]

[6] Gude N，Koponen T，Pettit J，Pfaff B，Casado M，McKeown N，Shenker S. NOX：Towards an operating system for networks. ACM SIGCOMM CCR，2008，38(3)：105-110. [doi：10.1145/1384609.1384625]

[7] Shenker S. The future of networking，and the past of protocols. In：Proc. of the Open Networking Summit. 2011. http://www.opennetsummit.org/archives/apr12/site/talks/shenker-tue.pdf.

[8] ONS. SDN：Transforming networking to accelerate business agility. 2013. http://www.opennetsummit.org/archives/mar14/site/why-sdn.html

[9] 舒文琼. SDN 领域两大组织角力 OpenFlow 协议为共性特征[J]. 通信世界，2013(25)：44-44.

[10] Bosshart P，Gibb G，Kim HS，Varghese G，McKeown N，Izzard M，Mujica F，Horowitz M. Forwarding metamorphosis：Fast programmable match-action processing in hardware for SDN. In：Proc. of the ACM SIGCOMM. 2013：99-110. [doi：10.1145/2486001.2486011]

[11] 张硕琳. Openflow 交换机在 NetFPGA10G 上的实现[C]. //2013 信息通信网技术业务发展研讨会论文集，2013：593-599.

[12] 刘旭山. 虚拟交换机 Open vSwitch 的核心态研究及硬件实现[D]. 北京：北京邮电大学，2013.

[13] 王小龙. 模块化分布式路由器数据平面研究与实现[D]. 北京：北京邮电大学，2010.

[14] Thomas N D，Gray K. SDN：Software Defined Networks[M]. 2013.

[15] 房秉毅，张歌，张云勇，等. 开源 SDN 控制器发展现状研究[J]. 邮电设计技术，2014(7)：29-36.

[16] Plugin development process[EB/OL]. [2015-05-06]. http://developer.huawei.com/cn/ict/Products/SDN/Components/Forum/NorthboundInterface

[17] 闫志坤. 盘点业界 SDN 控制器及 SDN 组织[J/OL]. http://net.it168.com/a2015/0122/1700/000001700914_1.shtml

第6章 基于SDN的异构网络融合技术

软件定义网络作为一种新型的网络架构，通过对资源进行集中式的管控，从根本上提高网络的资源调度能力，为实现异构网络融合提供了条件。本章介绍 SDN 在不同网络场景中的应用，并以异构无线网络为例，简要介绍网络融合方案。随后，针对网络能耗问题，介绍基于 SDN 的节能方案。最后，以 Overlay 技术为例，介绍了如何在当前已有的网络设备中部署 SDN。

6.1 异构网络融合引入 SDN 的优势

当前移动通信和互联网技术高速发展，信息获取和传输手段多种多样，数据存储和共享方式纷繁复杂，共同催生了多种业务类型[如 OTT (over the top，指通过互联网向用户提供各种应用服务)、云平台租用、移动支付等]并存的复杂、共生的异构融合网络。该网络包含了有线通信网络、宏蜂窝网络、微蜂窝网络、移动 Ad Hoc 网络及计算机网络等多种形态，相互之间呈现出一种共存演进的关系。

在传统的异构网络架构中，网络层和传输层保留了数据分组转发、分布式路由寻址等核心通信功能，同时为 QoS、VLAN(虚拟局域网)、流量控制等网络业务需求提供特定协议，从而保证了异构网络的互联互通。然而，由于网络控制与数据转发功能是以紧耦合的方式固化在网络设备之上的，协议之间的交互和协调变得越来越复杂，且对于业务的个性化需求要单独设置网络节点，从而进一步加剧了网络控制管理的复杂程度。因此，如何设计一种更加通用的、开放的体系结构，实现不同网络之间的互联互通，从根本上促进异构网络的融合成为当前通信网络研究的重点。

如图 6-1 所示，当前的异构融合技术都是基于一种“打补丁”的方式，这种模式下发展起来的异构融合网络已经成为一个规模臃肿、结构繁杂的系统，这在一定程度上增加了网络管控的复杂度，限制了网络运行的质量和效率，难以适应目前网络业务的发展。

软件定义网络(SDN)是一种新兴的、控制与转发分离并直接可编程的网络架构，主要应用在云数据中心、宽带传输网络等场景中，以集

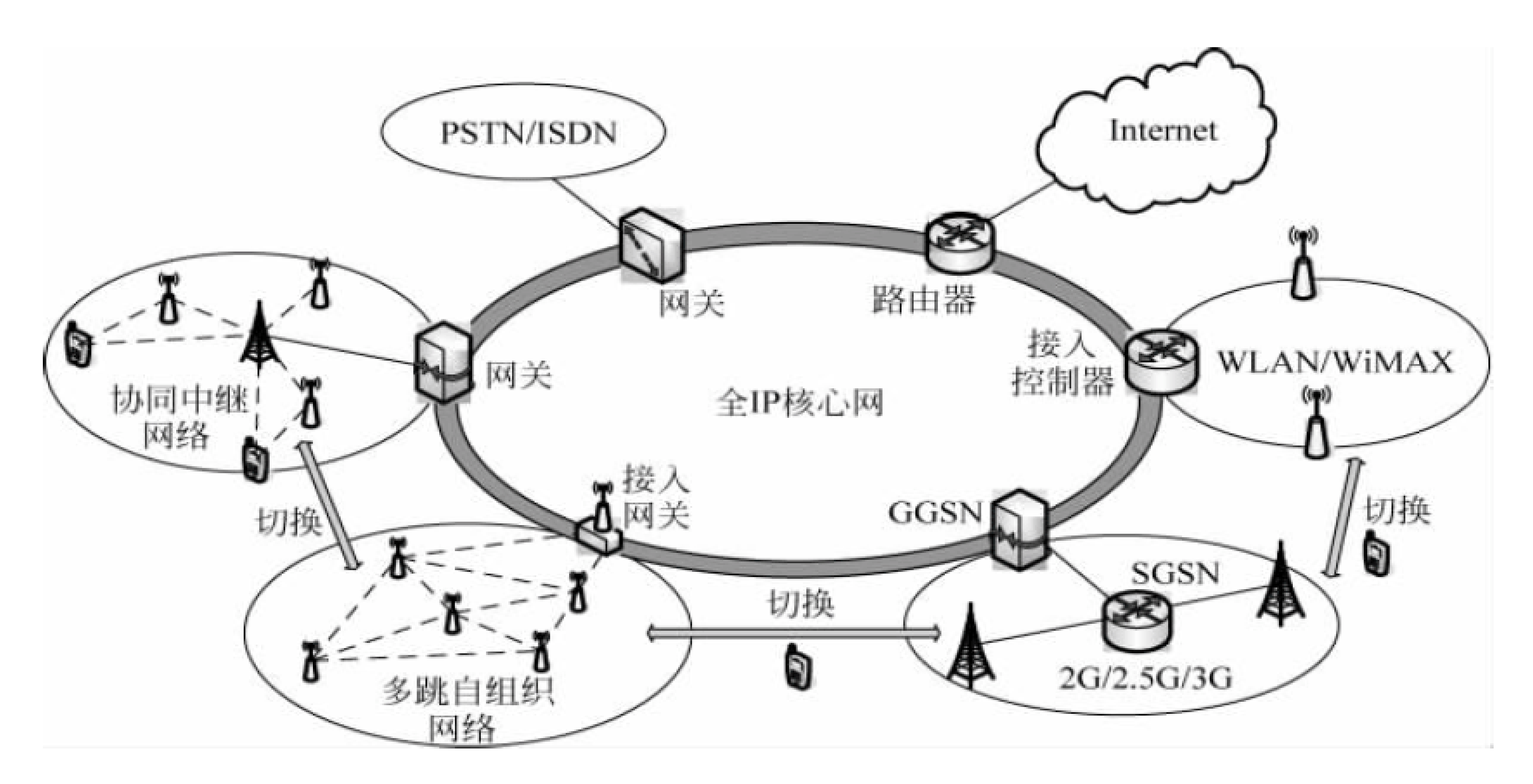


图 6-1　异构网络融合示意图

中的方式进行网络数据流的控制与管理,从根本上提高网络的资源调度能力。从本质上理解,SDN 是一个框架、一种网络设计的理念,可以将其引入以 IP 为核心的“小核心、大边缘”式的异构融合网络体系。

因此,将 SDN 应用到异构网络融合中将带来如下技术优势:

(1) 控制功能的解耦合,提高了异构网络中数据跨域传输能力。

当前的异构融合技术本质上主要依托 IP 骨干网通过隧道技术实现协议栈的改进,随着网络形态数量的增多,使得边缘网关承载的协议栈臃肿不堪。通过应用 SDN 思想,解离边缘网关的控制类协议,不仅简化了协议栈结构,还极大地释放了实体资源层设备转发数据的能力,为无处不在的云应用所产生的大量数据提供了高性能的传输保障。

(2) 实体资源统一封装,屏蔽了物理实体之间的异构性。

基于 SDN 思想,利用统一的资源描述语言对实体资源(如物理设备、链路资源等)进行抽象封装,能够有效隐藏底层网络资源的异构性,进而满足资源配置在可拓展性、安全性等方面的需求。当前,针对云计算的资源管理技术(如服务器虚拟化、存储虚拟化等)都已得到广泛应用,实现了底层实体资源的池化共享和按需分配。与此同时,针对路由器、交换机、无线中继等物理设备的虚拟化进程也在全面推进,为实现异构网络中的资源融合奠定了现实基础。

(3) 上层业务透明化,提高了业务类型的可拓展性。

异构网络中不仅存在实体资源的异构化,还存在业务类型的异构化,如会话类业务、交互类业务、流媒体业务等。在传统异构融合技术中,不同的业务将具有不同的 QoS 指标(如带宽、时延需求等),并对接入网络的用户终端有着不同的要求。基于 SDN 的思想,通过统一的北向接口对不同的业务类型进行特征分解,从而实现上层业务需求与底层实体资源的动态对接,提升业务类型的可拓展性。

6.2 SDN 应用场景简介

随着业务需求的改变,在现有网络架构下运营商所面临的挑战会越来越多,例如,接入点数量众多,运维工作量大,新业务拓展不够灵活,资源配置不够便捷等。面对这些挑战,运营商开始考虑将 SDN 技术引入到网络中。目前,运营商已经在多个场景中引入或即将引入 SDN 技术。

6.2.1 SDN 在数据中心网络中的应用

数据中心是一整套复杂的设施。它不仅仅包括计算机系统和其他与之配套的设备(如通信和存储系统),还包含冗余的数据通信连接、环境控制设备、监控设备以及各种安全装置。数据中心对于运营商网络的主要需求包括流量工程、网络资源高效管理、灵活组网(需要冗余保护)、虚拟机的快速部署及动态迁移、多租户支持与用户之间的互相隔离及 QoS 保障、端到端云服务,以及云数据中心互通。

目前 SDN 已经在数据中心实现了部署,而且相比于传统网络的管理控制机制,SDN 对于网络效率的提升更大。参考文献[1]详细描述了谷歌 B4 网络架构,在数据中心网络中,应用 SDN 使得链路利用率由 30%~40%提高到了接近 100%,证明了 SDN 在提高资源利用率方面具有巨大的应用前景。与 B4 系统基本原理类似,微软公司的 SWAN (software-driven WAN)系统[2]同样利用 SDN 体系结构实现数据中心间高效的利用率。它实现的手段是:当通过 SDN 全网信息观测到某条链路需求较低时,SWAN 控制数据层的数据通路迅速切换至该链路来传输数据,从而保证所有链路长时间的高效利用率。SDN 技术保障了 SWAN 能够进行全局观测以及流量工程的合理运用,确保资源利用率长期处于 60%以上。相对于 B4 系统,SWAN 系统采用的是传统设备,便于设备的更新与维护,更利于该系统的普及。

基于 Overlay 的 SDN 已有商用解决方案,通过在现有物理网络上叠加逻辑网络,可以实现网络虚拟化,满足云计算对网络灵活、动态、弹性配置等的众多需求。将 SDN 引入数据中心网络能够带来巨大的经济效益:第一,保护已有的设备资源,避免资源浪费;第二,提高网络的资源利用率,避免无限制地通过购买新设备来进行网络扩容,降低了网络建设成本;第三,支持业务创新,促进收入增长;第四,可以与云管理平台对接,降低运维难度。因此,将 SDN 应用在数据中心网络中,一方面可以为用户提供弹性的网络资源交付,另一方面还可以提供灵活、便捷的网络服务和端到端的云服务。

数据中心网络引入 SDN 后,主要的工作流程是:通过网关实现与传统网络第二层和第三层的互联,通过对物理网络进行虚拟化形成虚拟交换机,虚拟交换机利用 VxLAN/STT/NVGRE 的封装,在满足用户规模需求的前提下实现多租户隔离,控制平面与转发平面之间通过 OpenFlow 等协议进行交互,控制器实现对网络的集中式管理并进行资源调配,实现资源的优化利用,控制器之上与云平台实现对接,为用户提供灵活的端到端云服务,包括网络服务、计算服务和存储服务。

6.2.2 SDN 在家庭网络中的应用

家庭网络(home network)是融家庭控制网络和多媒体信息网络于一体的家庭信息化平台,是在家庭范围内实现信息设备、通信设备、娱乐设备、家用电器、自动化设备、照明设备、保安(监控)装置及水电气热设备等、家庭求助报警设备等的互连和管理,以及数据和多媒体信息共享的系统。

当前家庭网络中存在的问题主要有两点,一方面是家庭网络的终端设备众多,如家庭网关(home gateway,HG)、数字电视机顶盒(set-top Box,STB)等,维护升级困难,运维成本很高;另一方面,家庭网络中的终端设备业务耦合度高,部署新业务的周期很长。

因此,在家庭网络中引入 SDN 会带来众多便捷:第一,简化用户侧设备,减少运营商的投资和运维成本,节约能耗;第二,无须对家庭网关和数字电视机顶盒等进行持续的维护和升级,只需通过远程方式即可为用户提供网络故障诊断服务,便于故障诊断和修复,提升了业务可管理性,降低了 CAPEX(capital expenditure,资本性支出)和 OPEX(operating expense,运营成本)。

如图 6-2 所示,由于网络功能虚拟化(NFV)技术,今后家庭网关可以根植于通用的硬件平台,并在硬件上运行某种操作系统(如 Linux)。通过高性能的 DPI 实现 L2/L3 数据包解析及 L4～L7 数据包解析;通过 TR069 接口实现与网关管控接口对接以及与 ITMS(integrated terminal management system,终端综合管理系统)对接;通过 SDN 控制器接口实现与 SDN 控制器对接,接收控制器下发的流表及配置策略等信息,并上传拓扑信息、链路状态信息等。此外,为实现客户的自定义需求,可在家庭网关上加载运行相应的 APP,提供如动态分配带宽、QoS 保障等功能。

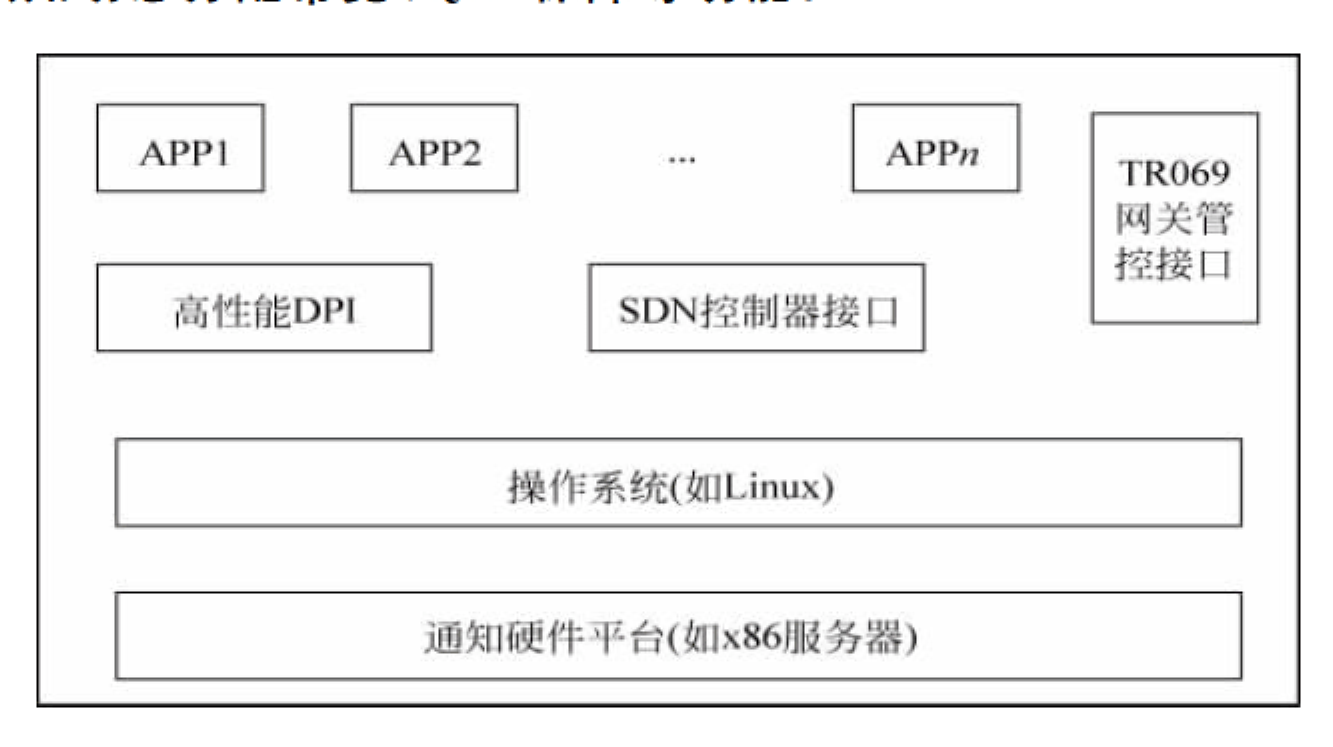


图 6-2 SDN 家庭网关原型

6.2.3 SDN 在城域网中的应用

典型的宽带 IP 城域网分成核心、汇聚以及接入三个层面,作为用户接入网络与核心骨干网络之间的桥梁,可以用来承载宽带、互联网专线、IDC(Internet data center,互联网数据中心)、移动互联网和其他(如 WLAN 及网管)不同的业务。典型的连接型 IP 城域

网架构如图 6-3 所示。为应对互联网长期以来采用网络不断扩容和粗放式经营模式造成的流量冲击，采用网络轻载以及扁平化建网策略，针对各类互联网为不同用户/业务提供不同 QoS 等级的差异化服务机制。

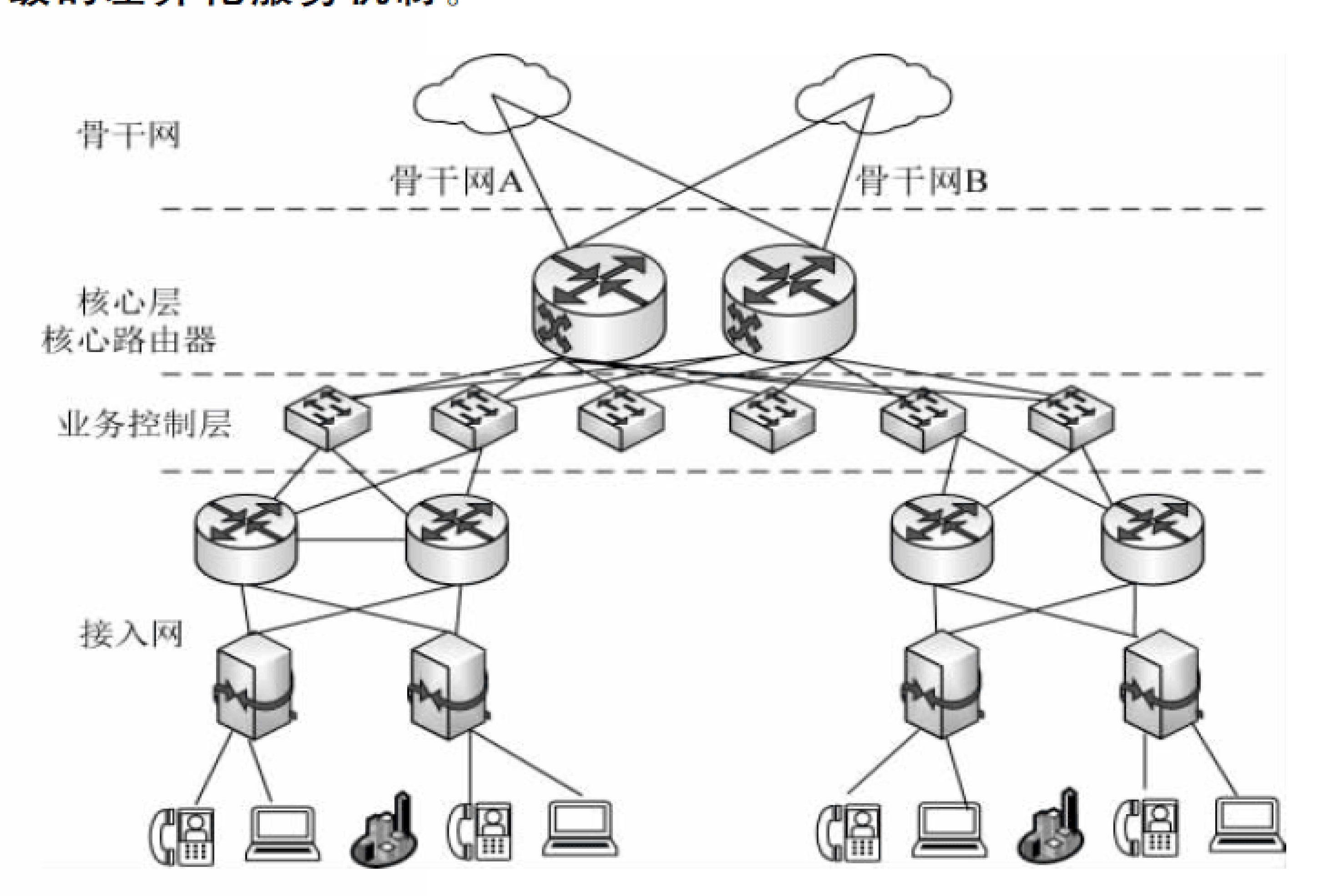


图 6-3　典型城域网架构图

随着互联网业务已经成为网络流量的主体，互联网业务从简单的网页浏览，不断丰富到高速下载、高清视频、在线游戏、电子商务、金融交易等各色各样的服务。业务类型不断细分，不同业务对网络质量要求差异化明显。当前粗放式经营的宽带 IP 城域网发展面临如下问题[3]：

（1）宽带接入及终端侧的问题。现有固定宽带业务与宽带接入方式以及接入终端之间强耦合，不但导致新业务部署复杂，而且家庭网络组网复杂，导致故障率高。为更方便的对接入及终端侧进行改进，运营商迫切希望实现宽带接入网络节点的可视、可管、可定位。

（2）城域汇聚及业务控制侧的问题。边缘设备主要根据地域用户覆盖采用分布式部署方式，各设备独立运行和维护，地址资源规划复杂，存在网络可靠性差、设备资源利用率低且不均衡、运维成本高等一系列问题。此外，边缘设备多为厂家封闭式系统，新业务上线慢，与业务系统对接复杂，支持新特性的设备软件版本开发周期长，新特性的部署周期长。

（3）城域核心侧的问题。城域网核心路由设备一方面对接骨干网业务资源，另一方面对接本地 IDC 业务资源，主要存在城域网核心路由器（CR）出口流量不均衡、大流量突发现象频发、部分本地化流量绕转到外网 IDC 访问等问题，无法实现针对业务资源分布以及用户访问行为的智能流量调度。

SDN 不仅能够实现对包转发的灵活控制，而且能够使更多的网络控制和策略方便地加载在 IP 城域边缘控制网络之上，使网络具有更高的智能和灵活性。

基于 SDN/NFV 的新型宽带 IP 城域网络具备以下几点特征[3]：

（1）超宽带。以下一代互联网为基础的“超宽带”网络，支持高速的光纤接入，实现局

部多业务大容量的边缘接入控制(大容量融合 BNG)功能以及大容量核心路由转发的快捷(100G/400G 集群)。

(2) 高智能。适应互联网业务体验保障的城域边缘"智能化"优化,匹配智能管道发展,增强网络能力,通过网络功能虚拟化(NFV)提供丰富的可扩展宽带服务能力,提升宽带价值。

(3) 简运维。采用节点融合以及集中远程配置手段,用户相关的业务属性由集中化的网络控制平面(SDN)进行端对端的配置和管理,实现统一配置、业务开通和软件升级,进一步提高网络效率。

(4) 大开放。引入网络协同和开放平台,实现对城域网各种资源以及网络能力的抽象和封装,为业务的开发提供开放的 API 接口,随着网络功能的丰富化,业务创新灵活性将极大地增加。

6.2.4 SDN 在接入网中的应用

所谓接入网,是指从骨干网络到用户终端之间的所有设备。其长度一般为几百米到几千米,因而被形象地称为"最后一公里"。由于骨干网一般采用光纤结构,传输速度快,因此,接入网便成为了整个网络系统的瓶颈。接入网的接入方式包括铜线(普通电话线)接入、光纤接入、光纤同轴电缆(有线电视电缆)混合接入和无线接入等几种方式。但是随着网络规模的增大,对接入网络方案也提出了更多的要求。例如,随着网络规模的增大,网络内节点数量的增加,使得运营商的运维工作量非常大,并且在现有网络上开发部署一个新的业务或者升级一个旧的业务的时候都比较复杂,导致新业务开发周期长等问题。而对于这些问题,SDN 架构的集中管理,以及开放灵活的网络编程接口的特点正好符合接入网的改变需求。因此运营商也试图将 SDN 技术引入到接入网来应对现在接入所面临的挑战。一个典型的应用场景如图 6-4 所示。

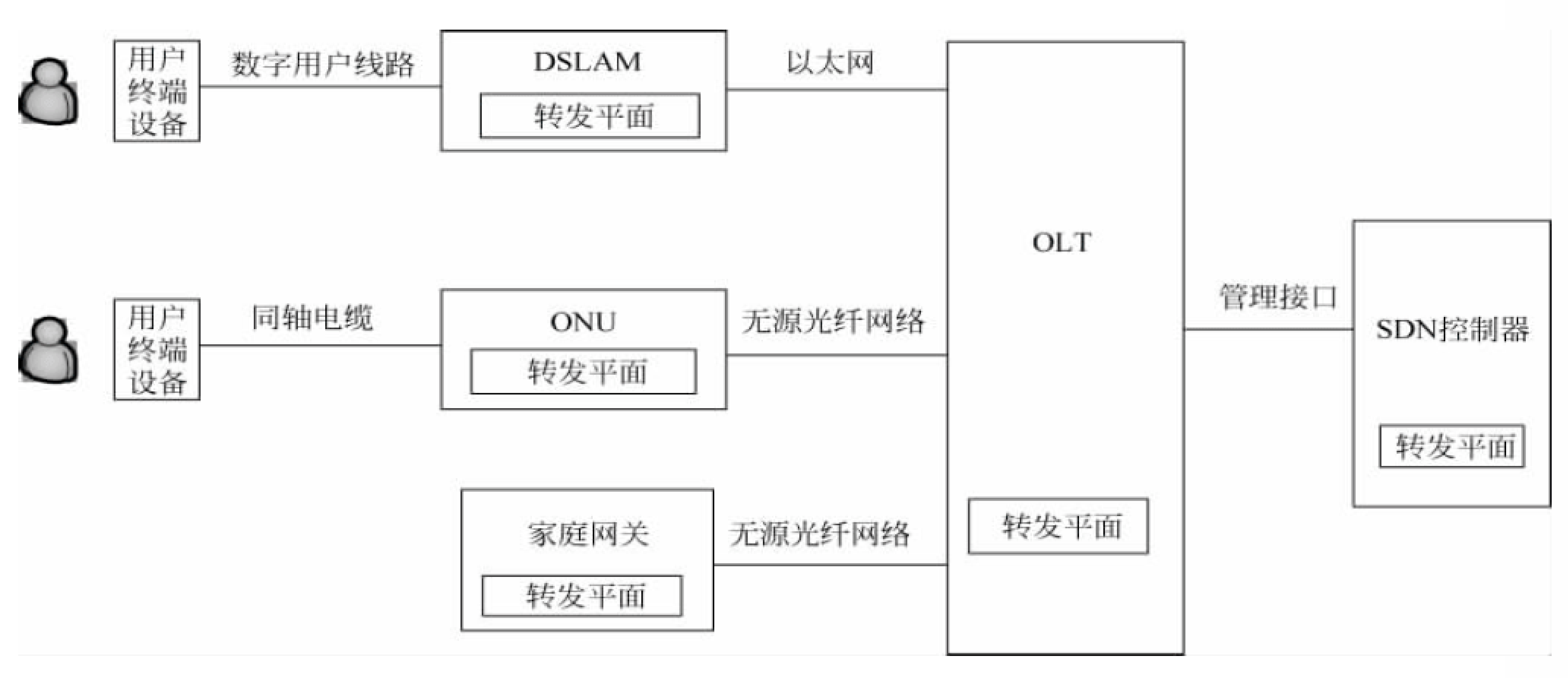


图 6-4 SDN 在接入网中的典型应用场景

根据 SDN 的设计理念，将数据转发层面和控制层面分离开来，在 OLT(optical line terminal，光线路终端)及其相关设备上保留数据面的相关转发功能，并为上层提供统一的编程接口，在控制层中实现这些设备的参数配置。当 SDN 控制器进行业务配置后，将配置参数下发，由 OLT 对配置参数进行内部分解，形成 OLT 及远端节点的配置参数并进行下发。通过这种方式，当有新业务部署时，只需要向节点下发新的流表信息即可，避免了大量的软件升级，加快了新业务的部署速度。如果控制器不能支持新业务特性，那么只需对 SDN 控制器进行升级，无须对已经部署的大量的硬件设备进行换代。另外，由于 SDN 提供了开放的管理/应用接口，这样就可以在不开放网管的情况下，让应用进行参数配置、状态查看等管理和维护操作。除此之外，还可以在虚拟接入网中引入分片能力，将一个虚拟接入网划分为多个分片，每个分片可以成为一个独立的逻辑接入网，包含指定的用户侧接口，并拥有虚拟接入网络部分或全部的业务能力。这样，用户可以实现自定义业务。

6.2.5 SDN 在 C-RAN 中的应用

随着信息社会的发展，端到端的通信已不再是当今网络流量的主导，通过网络获取以视频、网页等为主的信息已愈发流行。C-RAN(centralized-radio access network)现网条件，提出的新型无线接入网构架。C-RAN 是基于集中化处理(centralized processing)、协作式无线电(collaborative radio)和实时云计算构架(real-time cloud infrastructure)的绿色无线接入网构架(clean system)。其本质是通过减少基站机房数量，减少能耗，采用协作化、虚拟化技术，实现资源共享和动态调度，提高频谱效率，以达到低成本、高带宽和灵活度的运营。C-RAN 的总目标是解决移动互联网快速发展给运营商所带来的多方面挑战(能耗、建设和运维成本、频谱资源)，追求未来可持续的业务和利润增长。

传统的无线接入网具有以下特点：第一，每个基站连接若干固定数量的扇区天线，并覆盖小片区域，每个基站只能处理本小区收发信号；第二，系统的容量是干扰受限，各个基站独立工作已经很难增加频谱效率；第三，基站通常都是基于专有平台开发的"垂直解决方案"。这些特点带来了以下挑战：数量巨大的基站意味着高额的建设投资、站址配套、站址租赁以及维护费用，建设更多的基站意味着更多的资本开支和运营开支。此外，现有基站的实际利用率还是很低，网络的平均负载一般来说大大低于忙时负载，而不同的基站之间不能共享处理能力，也很难提高频谱效率。最后，专有的平台意味着移动运营商需要维护多个不兼容的平台，在扩容或者升级的时候也需要更高的成本。

而对于 C-RAN，射频接入单元(radio access unit，RAU)和基带处理单元(baseband processing unit，BBU)不再集中于单个小区，而是在每个小区放置射频拉远单元(remote radio heads，RRH)实现 RAU 的功能，将 BBU 集中起来形成一个云化的数据中心资源池。利用高带宽、低时延的链路将 RRH 和 BBU 连接起来。

如图 6-5 所示是一个典型的 SDN 应用架构。应用层有各种应用和服务，这些应用和服务主要来自于三类实体：内容提供商、网络服务提供商和设备制造商。其中，内容提供商既包括传统的内容提供商，又包括新兴的网络内容提供商(如亚马逊、谷歌等)。控制

层由一组实体分离但逻辑集中的控制器组成。一个控制器能控制一组网络实体，每个网络实体又能被多个控制器控制。核心网和接入网组成了转发层。转发层的基础设施经过虚拟化后被控制层控制。因此，通过控制器的策略调整，可以在同样的设备上灵活配置和自动部署不同的独立子网。下面对这三个层进行详细介绍：

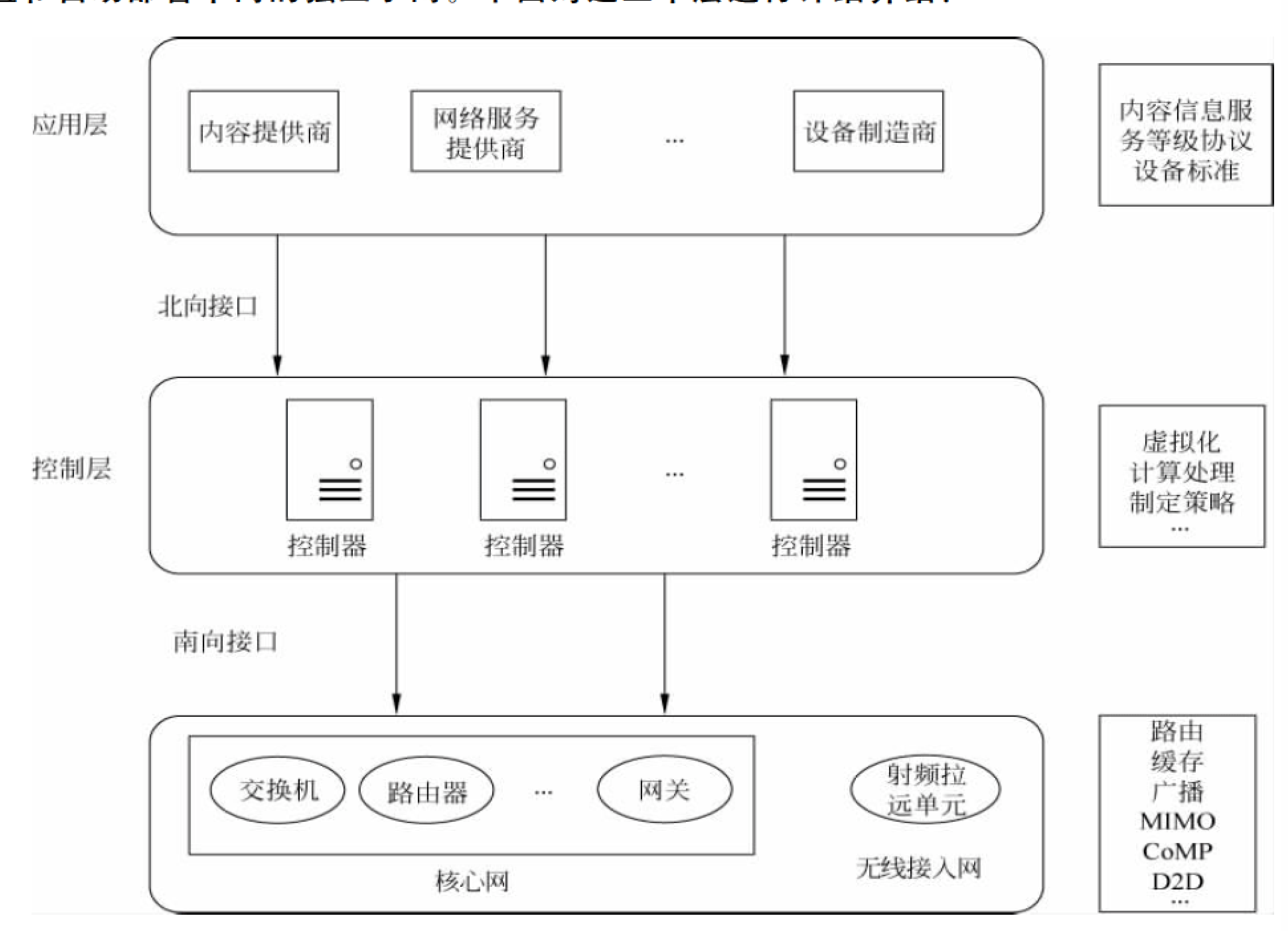


图 6-5 SDN 应用于 C-RAN 中的架构

(1) 应用层。内容提供商通过网络服务提供商将指定数据对象发送给用户。由于 ICN(information-centric networking，信息中心网络)利用指定数据对象(NDO)将信息与位置解耦，因此内容提供商可以更好地分析和预测用户的行为。结合不同的网络服务提供商所能提供的服务等级，内容提供商选择最佳的网络进行服务。通过 SDN 将硬件与软件解耦，使得内容提供商与网络服务提供商能够摆脱设备制造商的瓶颈。应用层的需求可以通过开放的 API 接口传递给下面的控制层，利用控制器中的协议实现。

(2) 控制层。在网络操作系统中控制层是核心部分，在控制器上可以实现基础设施虚拟化、可编程抽象，甚至内容命名、寻址和匹配操作。控制器利用对系统全局信息的掌控，通过网络操作系统对网络进行配置和管理，以此更好地利用资源。

(3) 转发层。因为控制功能已经被抽离并集成到了控制层，转发层由只需提供简单信息交换和转发的虚拟网络设备组成。但是，一个典型的异构网络中可能有上千个设备，大规模控制可能会造成单个控制器过载，因此需要设计分布式的控制单元。

6.2.6 SDN 在 VPN 中的应用

虚拟专用网络(virtual private network,VPN)的功能是:在公用网络上建立专用网络,进行加密通信。VPN 在企业网络中有广泛应用。VPN 网关通过对数据包的加密和数据包目标地址的转换实现远程访问。VPN 有多种分类方式,主要是按协议进行分类。VPN 可通过服务器、硬件、软件等多种方式实现。VPN 属于远程访问技术,简单地说就是利用公用网络架设专用网络。例如,某公司员工出差到外地,他想访问企业内网的服务器资源,这种访问就属于远程访问。

让外地员工访问到内网资源,利用 VPN 的解决方法就是在内网中架设一台 VPN 服务器。外地员工在当地连上互联网后,通过互联网连接 VPN 服务器,然后通过 VPN 服务器进入企业内网。为了保证数据安全,VPN 服务器和客户机之间的通信数据都进行了加密处理。有了数据加密,就可以认为数据是在一条专用的数据链路上进行安全传输,就如同专门架设了一个专用网络一样,但实际上 VPN 使用的是互联网上的公用链路,因此 VPN 称为虚拟专用网络,其实质就是利用加密技术在公网上封装出一个数据通信隧道。有了 VPN 技术,用户无论在外地出差还是在家中办公,只要能上互联网就能利用 VPN 访问内网资源,这就是 VPN 在企业中应用得如此广泛的原因。

现在各大企业对于 VPN 的需求越来越大,然后当前网络中,企业 VPN 业务存在着诸多问题。首先,对于用户而言,申请 VPN 的成本非常高。其次,VPN 一旦部署完成,就是一个静态链路,无法满足客户灵活多变的业务需求,而对于运营商,每部署一条 VPN,新业务、新功能引入成本升高,部署周期也变长。此外,VPN 端到端的配置相当复杂,需要逐业务、逐设备进行配置,而且运行维护很困难。

利用 SDN 技术,可以通过控制器全程配置,简化业务策略配置工作,缩短业务的部署时间;还可以无缝对接私有云/混合云服务,提升客户的价值体验。将 SDN 应用到 VPN 中将带来如下便利:

(1) 从技术角度,基于 OpenFlow 的软件定义 VPN 已有厂家解决方案,证明了其可行性;SDN VPN 可作为现有 IP/MPLS VPN 在客户端的延伸,简化部署;针对私有云或混合云客户,控制器范围可控。

(2) 从经济层面,主要有 4 个方面的价值。第一,保留了已经部署的设备,避免了设备资产的浪费;第二,VPN 的部署可以全程通过软件自动化配置,只需一次性定义,按需连接部署,降低了运行维护成本;第三,对设备进行了简化,降低了硬件成本;第四,为传统业务提供了新的服务方式,将云服务延伸到企业级。

(3) 对于客户而言,一方面可以灵活高效地为用户提供按需的 VPN 服务,提升了用户体验;另一方面,降低了客户端设备的投资成本。

如图 6-6 所示,在部署了 SDN 之后,企业客户不需要再购买 CPE(customer premise equipment,客户终端设备),简化为 OpenFlow 交换机或者通用的服务器,这些交换机或服务器作为 IP/MPLS 网络的延伸,连接到各自的数据中心,实现数据中心与企业 IT 互联。交换机、数据中心以及 IP/MPLS 网络均连接到 SDN 控制器,受到控制器的集中控

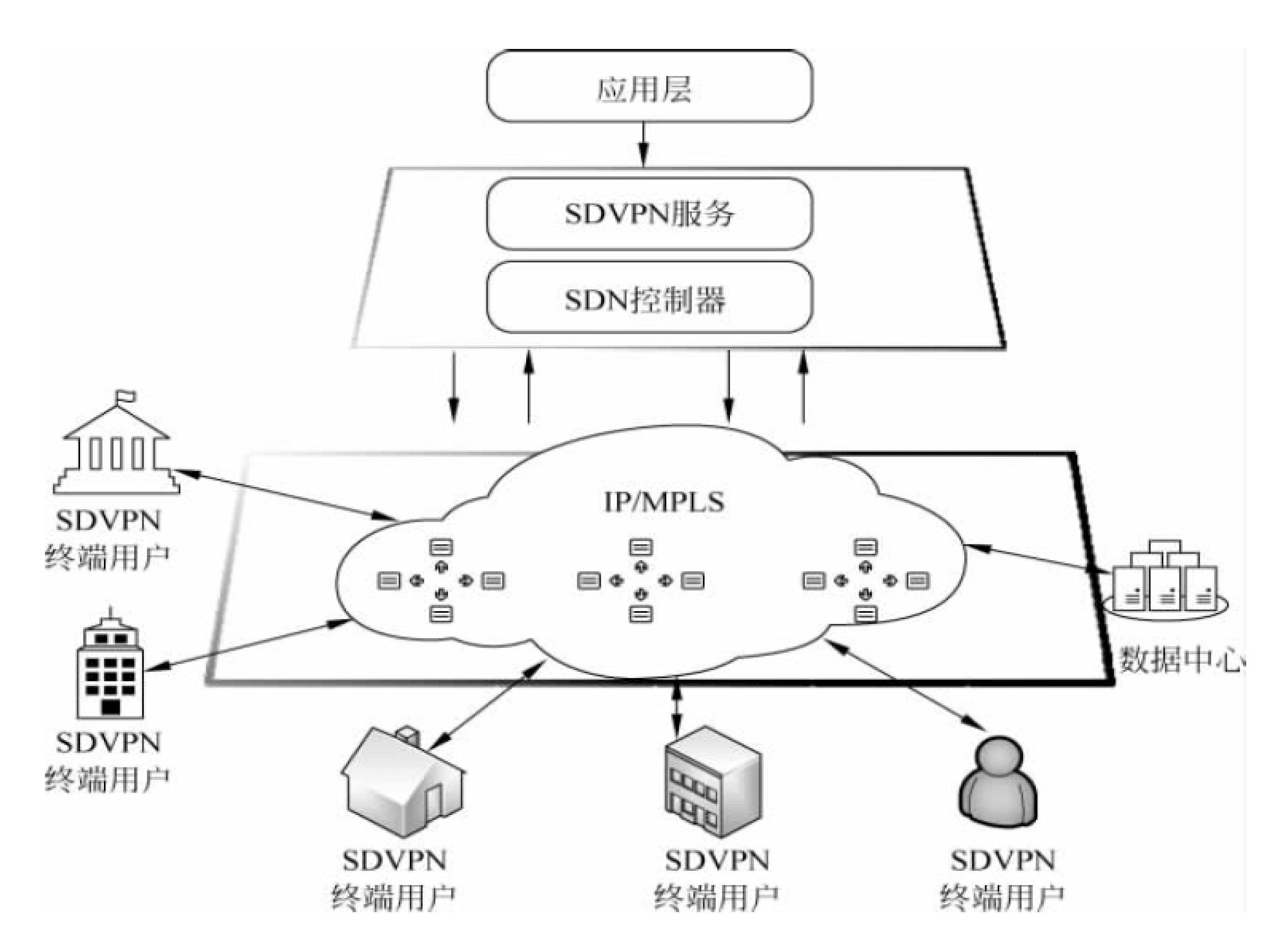


图 6-6 SDN 应用于 VPN 中

制，控制器收到应用层的定义设置，进行自动化配置，实现 VPN 网络的一次定义、按需连接。

6.2.7 SDN 在固定移动融合场景中的应用

固定移动网络融合(fixed mobile convergence，FMC)就是通过固定网络与移动网络之间的融通、合作，从而实现全业务及融合业务的经营。随着互联网在人们日常生活中的重要性不断增加，人们要求随时随地接入网络的需求持续增长。各种差异性的业务，包括互联网业务，如 Web 访问、视频等，以及运营商自有业务，都可以通过 Wi-Fi 连入固网，也可以通过无线网络接入[4]。接入层面的固定移动融合正逐步得到体现。

然而，处于接入层面与承载层面之间的汇聚网关仍然是有线网络和无线网络独立设置的。由于 2G/3G 阶段无线网络的汇聚流量相比于有线网络仍然有限，无线网络的汇聚网关 SGSN/GGSN 与有线网络的汇聚网关 BRAS 虽然功能类似，但是网络层级的位置并不对等，因此二者难以融合。

当固定移动网关在网络层级上处于较为接近的状态，且经过控制转发分离，则固定移动网关设备的融合成为可能。如图 6-7 所示是基于 SDN 的固定移动融合网关远景示意图。基于 SDN 的固定移动网关可以采用共同的通用转发设备。根据需要，如安全性、多租户等，对网络流量进行隔离。不同网络的应用则完全基于 SDN 控制器来执行，从而在控制面融合了固定移动网关。固定移动网关的不同功能差异由应用差异来体现，而不是由设备差异体现。基于 SDN 的固定移动网关融合从降低 CPAEX、提高网络资源利用率和网关设备的可靠性等方面为运营商带来收益。

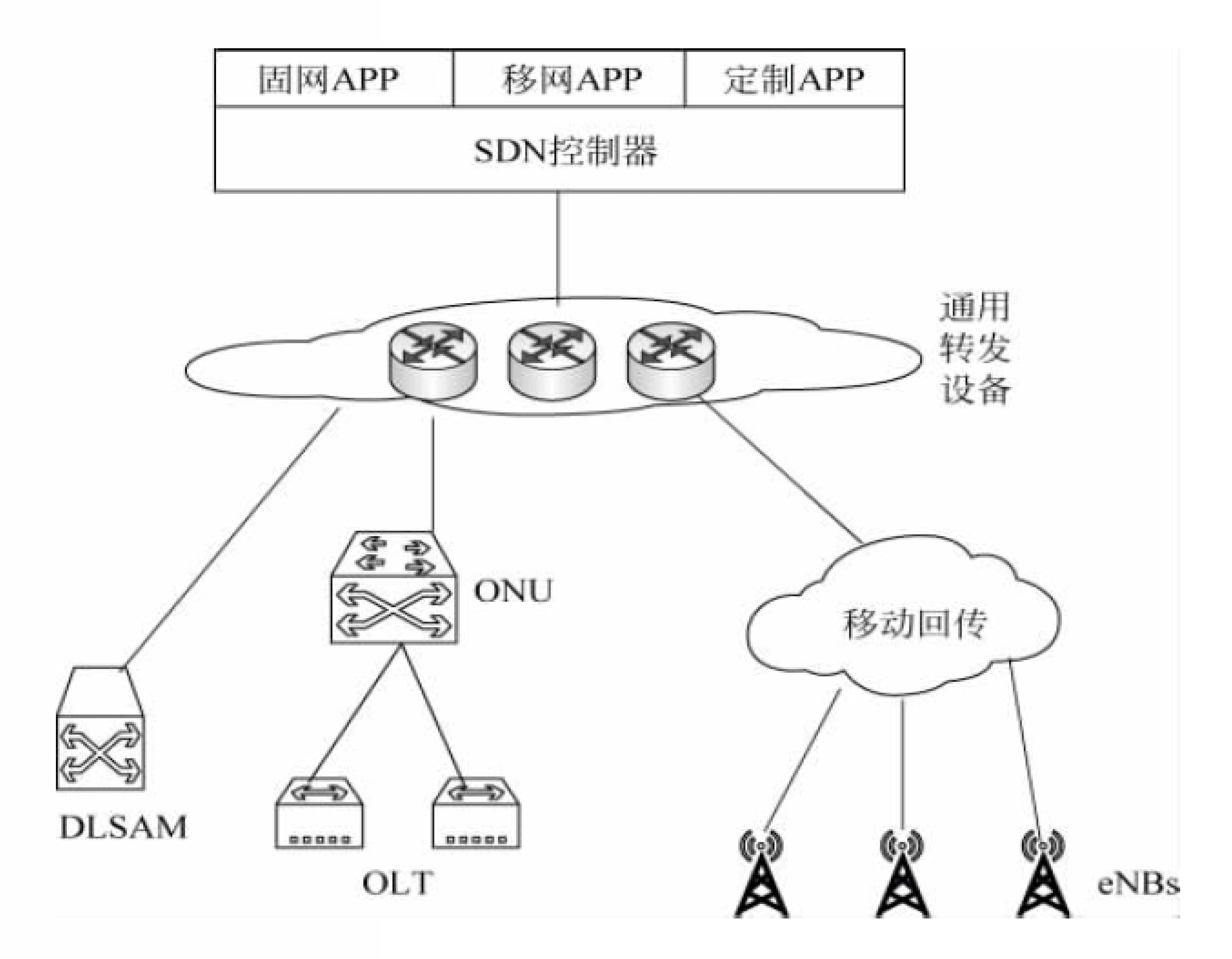


图 6-7 基于 SDN 的固定移动融合网关场景

6.3 基于 SDN 的异构无线网络资源管理方案

以智能手机为代表的智能设备为用户提供了日益增长、种类繁多的各式应用，用户对于通过无线网络随时随地获取数据和多媒体服务的需求急剧增长。随着无线通信系统的快速发展，人们已经习惯并且依赖无处不在、无时不通的无线网络。特别是随着物联网技术的广泛应用，智慧地球、智慧城市等概念的提出，使得对通信系统尤其是无线通信系统作为一种城市、社会基础服务的需求更加迫切。

传统依靠新型的无线信道技术和有限的带宽扩展无线蜂窝系统的做法，已经远远不能满足当前无线流量爆炸式增长的需求。根据库伯定律（Cooper's law）的预测，通信系统中无线设备的数据需求大约每 30 个月增长 1 倍。2014 年 5 月 Cisco 发布的全球移动流量数据报告显示，2013 年全球移动数据流量比 2012 年增长了 81%，并且预测移动流量在 2013 年至 2018 年间增长近 11 倍[5]。

用有限的频谱资源来尽量满足用户无限增长的业务需求是无线通信系统的基本目标，这需要尽可能地提高频谱资源的利用效率，同时降低由于频率复用而带来的设备间干扰。随着无线传输技术的快速发展以及智能设备的广泛应用，用户对于无线业务的速率需求永无止境。尽管 3G、4G 技术通过改进编码技术、信号模式及收发端的物理层技术等为无线通信系统的频谱利用效率带来了 2～5 倍的提升，但仍然无法满足用户呈指数增长的数据需求。因此，如何进行无线网络资源管理以提高资源的利用率是未来网络研究的重点问题。

6.3.1 网络资源管理简介

在异构无线网络环境中，用户并不能根据各网络的实际频率资源和转发资源接入合

适的网络,网络资源利用率往往很低,基于 SDN 的异构无线网络资源管理主要包括两个内容,一是针对传统无线网络的资源管理,二是针对虚拟化场景中的虚拟资源管理。

1. 传统资源管理

传统资源管理根据网络中存在的不同资源类型,可以分为 4 个方面:功率资源、时间资源、频率资源和空间资源。

(1) 功率资源。功率控制是提高网络能量效率的一个重要途径。根据使用场景不同,功率分配可以体现为单用户时延约束下的功率分配、单用户多载波网络子信道功率分配、MIMO 信道功率分配、多用户网络干扰最小化的功率分配、多节点联合传输功率分配、中继节点功率分配。

(2) 时间资源。由于无线环境的时变性和数据包到达的随机性,设备中缓存的数据量是不断变化的。为了增强能量效率,需要根据不同业务的时延约束,对数据包进行合理调度。在 TDMA(time division multiple access,时分多址)网络中,为了节能,每个用户都试图增大自身的传输时间。因此需要为每个用户设计合理的传输时隙个数,联合优化 MAC 层和物理层参数。

(3) 频率资源。根据香农公式,在单用户传输场景下,增加系统带宽可以提高能量效率。而在 OFDMA(orthogonal frequency division multiple Access,正交频分复用多址接入)系统中,多个用户共享频率资源,增大某个用户的传输带宽,会导致其他用户的可用带宽减少,因此,需要为每个用户分配最优载频个数。

(4) 空间资源。利用多点联合传输技术,构成虚拟 MIMO 系统,可以提高能量效率,并为通信系统带来空间自由度。但是多点联合传输会增加额外的信令开销,增大传输时延。另外,从香农公式可知,数据传输所需能量随着距离增大呈指数增长,采用多点作为转发媒介点可降低能耗,但同时会增加传输时延,而且中继也会产生固定能耗。因此,需要选择合适的中继节点来提高网络能量效率。

2. 虚拟资源管理

对于异构无线网络而言,异构性不仅存在于实体资源层,同时也体现在用户业务类型方面,这就造成了资源调度困难,难以实现资源效率与管理效率的最大化。针对这些问题,运营商提出虚拟化技术将硬件资源虚拟化,并将资源分为虚拟计算资源、虚拟存储资源和虚拟网络资源等,运营商通过软件来集中管理这些虚拟资源,以最大化实体资源利用率。

网络虚拟化环境屏蔽了底层网络的差异性,底层实体网络在类型和地理位置上的差异导致了虚拟网络在统一管理上的差异。虚拟化环境中资源具有如下特点:

(1) 异构性与分布性。网络虚拟化环境中的底层网络资源包括节点资源和链路资源。节点资源主要包括路由器、交换机和无线网络基站等设备,这些设备在功能、配置、操作系统以及访问方式等方面存在很大的差别;链路资源主要包括光纤、双绞线和无线链路,在通信方式、性能(带宽、时延和吞吐量等)等方面也各不相同。节点资源和链路资源的多样性导致了虚拟资源的差异性,进而给虚拟资源管理带来了困难。此外,由于底层网络是通过不同的基础设施提供商构建和部署的,而且由于不同地理位置的物理网络

导致的虚拟资源的分布性，实体资源的差异性和分布式特性更为虚拟化的调配和统一管理增加了难度。因此，需要屏蔽底层资源的异构性以及分布性，以简化虚拟资源的管理。

（2）独立性和自治性。作为底层网络的拥有者，基础设施提供商通常独自使用自己的底层资源，并对底层设备进行自主管理与维护。因此，需要按照一定规则统一命名、管理和配置底层资源。

（3）动态性和可扩展性。通过网络虚拟化，服务提供商可以根据用户需求动态地调整所占用的虚拟资源，也可以根据底层网络的实际情况动态地占用或更换某个底层网络设备。尤其是当底层网络为无线网络时，信道状况、资源的位置和负载量一直处于动态变化中，因此需要在资源动态变化的情况下保证虚拟资源的稳定性，维护虚拟网络拓扑。可扩展性主要包括两个方面：一方面是基础设施提供商会增加新的底层设备，用来扩大网络规模；另一方面，用户会提出新的虚拟网络需求，这就要求服务提供商能够对现有虚拟网络进行扩容。因此，虚拟资源管理系统需要具有可扩展性，以适应底层网络资源和用户虚拟网络需求的变化。

（4）虚拟资源请求的限定性和约束性。服务提供商根据用户的需求向底层网络请求资源，可选择的资源包括节点资源（如 CPU、存储空间等）和链路资源（带宽、时延等），更高级的可以选择限制底层网络结构和底层网络拓扑等。但由于实体网络资源的数量有限，这就对全网虚拟请求总量造成了约束。

综上所述，网络虚拟化环境中的资源管理位于底层物理资源和服务提供商之间，通过屏蔽底层物理资源的异构性、动态性和分布性，为服务提供商提供一个资源访问接口，呈现在服务提供商面前的是一个抽象的资源池。

通过对传统的业务和设备之间的解耦，虚拟资源管理构建一种资源可共享、节点可重构、能够提供柔性服务的网络，有利于互联网转型为集转发、存储和计算功能于一体的智能网络。服务提供商根据不同的业务特性和虚拟网络中用户数量、网络拓扑、带宽等需求按需租用物理资源，提供满足业务需求的自定制网络，从而实现网络的“个性服务”。通过对网络资源进行集中管理，促进异构网络融合。

6.3.2 频谱资源分配

频谱资源管理主要有两个目标：

（1）提升频谱利用率。近年来，用户业务呈现应用多样化、需求复杂化的趋势，而移动通信系统中的频谱资源严格受限，制约了移动通信的发展，因此，如何提高频谱利用率一直是移动通信系统的重点研究内容。与 3G 相比，现有的 4G 移动通信网络频谱效率提升明显，然而，用户日益增长的数据速率需求和有限的频谱资源之间的矛盾仍未得到很好的解决，因此，在满足分组多业务服务质量需求的基础上，提高移动通信网络容量和频谱利用率是移动通信系统中频谱资源管理的目标之一。

（2）抑制干扰。移动通信系统中通信环境复杂多变，为了提高频谱资源利用率，近年来的多种新型组网模式多采用同频复用技术，这种复用技术的缺陷之一是共信道干扰。因此，如何解决共信道间干扰问题，是提高网络性能的关键。

利用 SDN 集中管理的特点，可以通过控制层对频谱资源进行更加优化的分配，如低

负载小区载频板动态关闭、对休眠基站中载频板的频谱进行动态重新分配、高负载小区动态增加可用频谱等方案，使得用户能够动态使用更多更好的频谱来进行数据传输，从而增加了网络的吞吐量，提升了用户体验。

此外，由于无线SDN网络中，转发设备需要根据控制器下发的转发规则进行操作，为了控制信息能实时更新，需要保证控制信息的优先转发，因此需要设计相关的优先级策略，为控制信息分配固定频段或者在与数据信息共享频谱时给予控制信息更高的优先级。下面介绍在无线Mesh组网架构中关于控制流和数据流所占频谱资源分配的解决方法。

关于控制信息的传输，目前主流的解决方案分为out-of-band和in-band两种。out-of-band部署有线网络来传输控制信息，但是在实际应用中，由于需要额外部署线路，往往会比较困难。相对来说，in-band控制信息传输不需要额外的基础设施，控制信息像瀑布一样从控制器流到各个交换机中。

在有线网络中，控制流和数据流可以通过专用的线路传输，但是对于无线网络，如图6-8所示，数据流和控制流要共享无线资源。尽管无线Mesh网络中多信道多无线电技术已经被用在信道分配中，但是由于硬件技术的限制，一个天线接口在同一时间只能使用一个信道，并且一个路由器所能同时占用的信道数要受到该设备总天线数的限制。

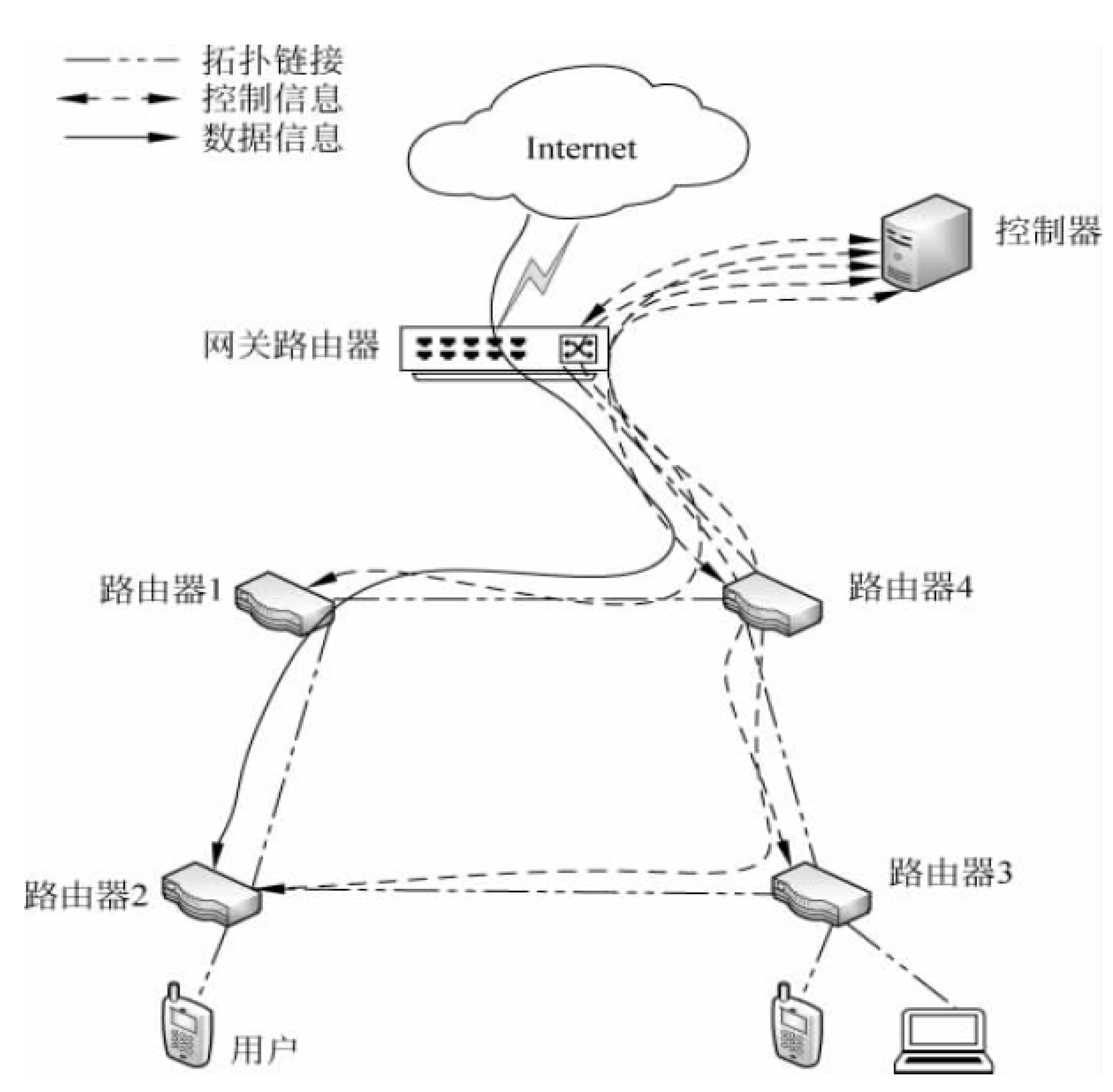


图6-8 基于SDN的无线Mesh网络拓扑示意图

控制信息主要包括状态信息和流表规则信息，其中状态信息主要包括每条链路的速率、每个转发设备的规则大小、每个无线电端口的频谱分配情况等。当控制流与数据流竞争无线频谱资源时，如果没有合适的竞争机制，会导致网络性能降低和资源浪费。因

此，需要给控制流更高的优先级。

如图 6-9 所示是一个 SDN 无线 Mesh 路由器的结构图。相比于传统路由器，该路由器增加了监控模块，能将连接状态和邻居信息通过 OpenFlow 协议发送给控制器，控制器收到底层信息后，结合北向接口传来的应用层需求，做出各项决策，如路径计算、流量调度、频谱分配等，并将控制信息通过南向接口发送给路由器，路由器根据收到的规则进行转发。

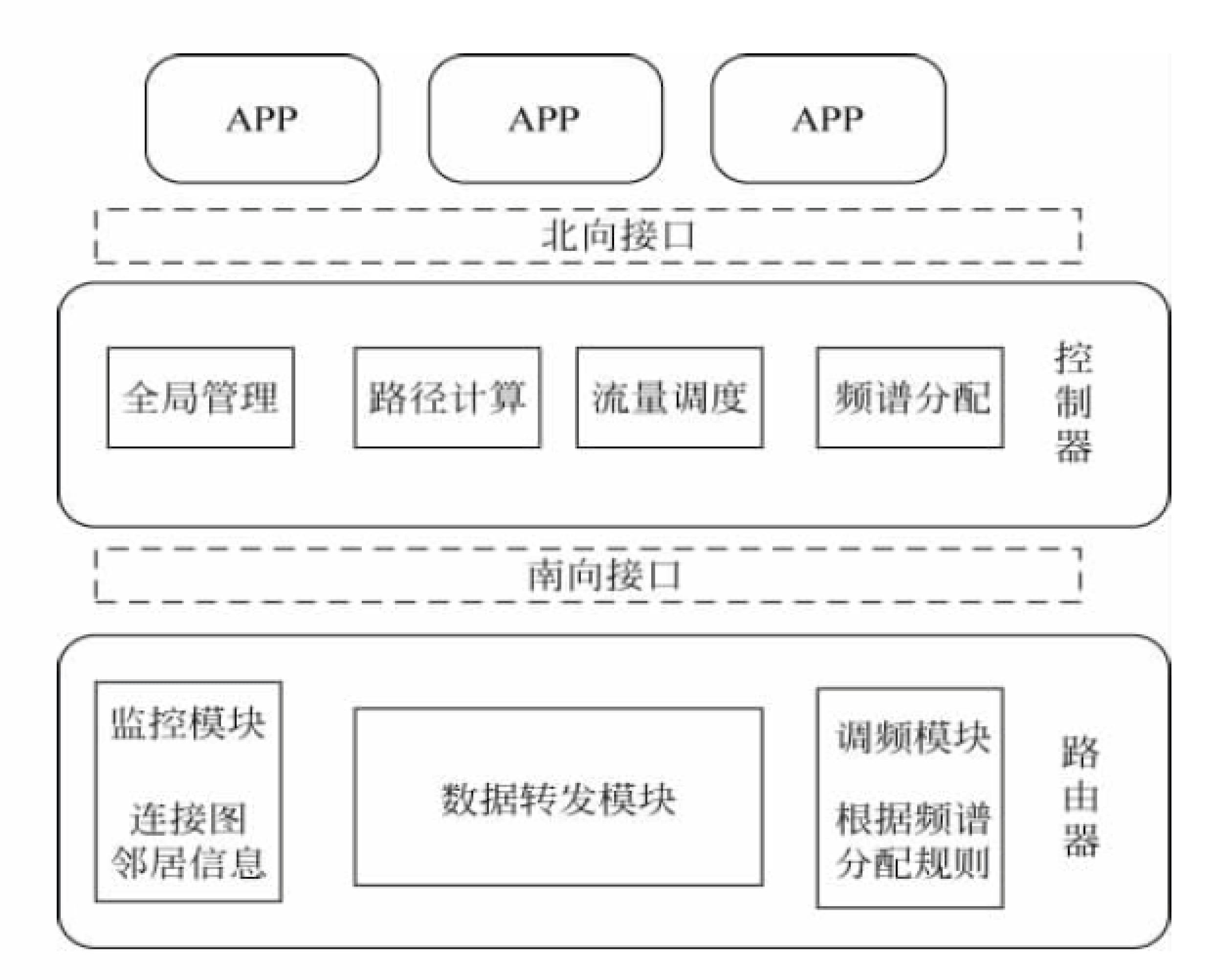


图 6-9　SDN 无线 Mesh 路由器结构图

假设只有目的接收机在传输范围内且不受其他不相干发送器的干扰时，才能成功传输，换句话说，如果一个接收节点位于一个非计划发送节点的传输范围内，就认为这个节点会受到干扰，那么接收节点就不能从源发射节点收到信息。如图 6-10 所示，1 给 4 发送，占用频段 B1，那么由于 1 是发送者，而且接收节点 4 处于 5 的传输范围之内，因此，链路(1,2)、(2,1)、(4,1)、(4,5)、(5,4)都不能使用频段 B1。

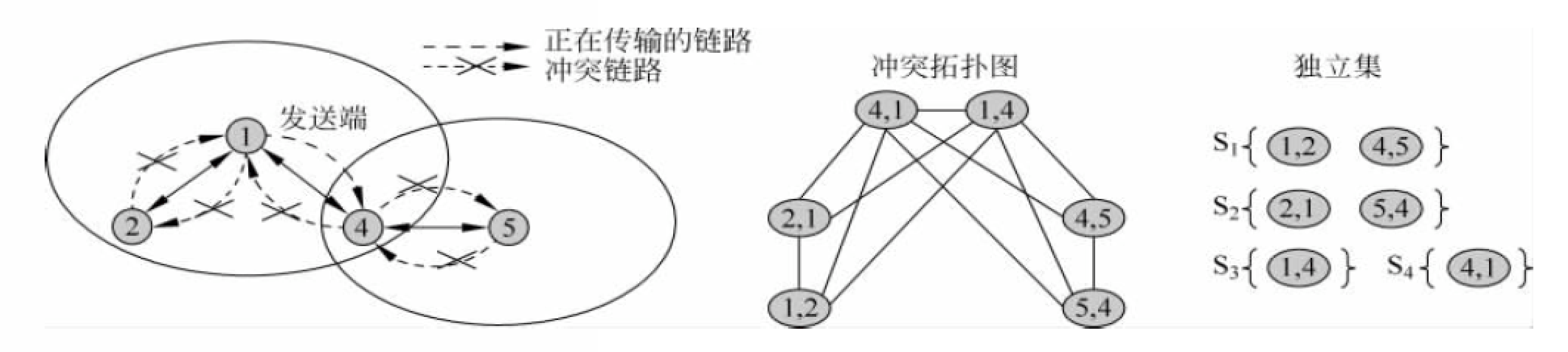


图 6-10　由无线拓扑图构建独立集

因此，可以先构造出冲突图，即链路之间如果在图中相连，则不能同时共用同一频段，由此可得出独立集，即处于同一独立集中的链路可以同时共用同一频段。因此，对独立集所传输的控制信息和数据信息进行加权统计，并按大小进行排序，依次给每个独立集分配一个频段，循环结束后判断是否还有可用频段，若有则再次进行分配，直到全部分配完成。在频段分配完成之后，为了保证控制信息的优先性，提出了固定频段不共享算

法、不固定频段不共享算法以及不固定频段共享算法。

(1) 固定频段不共享算法：在频段分配完成之后，将某几个频段分配给控制信息，剩余频段分配给数据信息，不能占用其余频段进行传输。

(2) 不固定频段不共享算法：由于流量分布的随机性，导致固定频段不共享算法的频谱利用率很低。例如，对于离控制器比较近的链路，需要传输的控制信息多；而对于离网关比较近的链路，需要传输的数据信息较多。为了避免在繁忙时链路出现拥塞，提出了不固定频段不共享算法。控制信息和数据信息可以自由选择可用的频段，但是控制信息有更高的优先级。

(3) 不固定频段共享算法：有时控制数据所占带宽甚至会小于某一频段的容量，这就会使得某些频段的承载能力没能完全利用。为了进一步提高频谱资源的利用率，在满足控制流之后，如果在每个子频带仍有可用带宽，剩余带宽用来传输数据流。

6.3.3 转发资源分配

无线网络中转发资源的分配主要包括两种技术，一是虚拟化技术，二是切片技术。通过虚拟化屏蔽了资源底层差异，通过切片技术灵活构建虚拟子网，能更好地利用物理资源。

1. 虚拟化技术

资源管理的本质是为了提高资源的利用率。网络虚拟化环境中资源管理的目的也不例外，即在满足用户需求的前提下，尽可能地提高底层物理资源的利用率。为此，首先需要资源发现，对可用资源进行标记，然后进行资源探测，测量出具体可用资源量，最后才能进行资源管理。虚拟化环境中的资源一般分为节点资源和链路资源。节点资源包括计算资源、存储资源和转发资源等，链路资源主要是指链路的带宽和时延等。

(1) 资源发现

资源描述是指通过对底层物理资源做标记，使得该资源能够被发现和访问。通过对资源类型进行区分，如节点资源和链路资源，再通过资源发现对其进行定量描述。

(2) 资源探测

链路带宽资源发现一般采用主动测量的方法。根据探测的方式，带宽测量模型分为探测速率模型(path rate model，PRM)和探测间隔模型(probe gap model，PGM)。

基于自感应拥塞的探测速率模型的基本原理是：在发送端以不同的速率发送探测分组，在接收端观察探测分组的时延，通过判断拥塞的初始时刻，推算出可用带宽。PRM的测量工具有TOPP、Pathload、PTR、Pathchirp等。

探测间隔模型则在发送端发送两个连续的分组，通过分析发送间隔与到达接收端的时间间隔关系估计可用带宽，但该模型需要事先获取路径的容量。PGM的测量工具有IGI、Delphi、spruce等。

节点资源发现即探测每个节点中剩余的可用CPU和内存等资源。目前主要的虚拟化技术都可以通过管理程序获取每个虚拟系统的CPU资源使用情况。

(3) 资源管理

通过虚拟化技术，可以在物理网络上抽象出多个相互独立的虚拟网络，服务提供商根据用户业务需求的特点为每个虚拟网络单独制定专用的网络协议和数据转发方式。

基础设施提供商向服务提供商出售虚拟网络资源，因此，对基础设施提供商来说，如何分配和调度虚拟资源是利益最大化的关键；而服务提供商向用户提供网络服务，因此，对服务提供商来说，如何根据不同用户的需求来租用适量的虚拟资源是利益最大化的关键。

2. 切片技术

当前学术界对软件定义网络的虚拟化问题有多种设计思路，其区别主要体现在“转换单元”位置上。根据实现虚拟网络资源到物理实体资源映射方式的差异，共有 5 种 SDN 虚拟化的设计思路[7]。

(1) 转换单元作为一个独立的外置设备，工作在交换机和控制器之间，为交换机和控制器之间的信令交互作代理，根据虚拟网络的配置，把不同的网络控制信令交付给正确的控制器和交换机处理。

(2) 转换单元集成于交换机中，交换机同时连接多个控制器，并根据交换机上的虚拟网络配置信息把流量交付给相应的控制器处理。

(3) 转换单元也是集成于交换机中，但是与方案 2 不同的是，每个交换机可以运行多个 OpenFlow 实例，每个实例连接一个控制器，转换单元根据虚拟网络配置信息把流量交付给相应的 OpenFlow 实例。

(4) 与以上方案的不同点在于，交换机把数据转发平面也进行平行分割，转换单元通过识别不同的转发端口(包括虚拟端口和物理端口)与分割后的转发平面对应。

(5) 方案 1 和方案 3 的综合，利用两个转换单元协作完成网络虚拟化，一个转换单元负责数据平面的虚拟化，另一个作为交换机与控制器之间的协议代理。

FlowVisor 就是基于第一种方案设计的网络虚拟化工具，即在数据平面与控制平面之间添加一个虚拟化层，通过虚拟化层控制各个虚拟网络逻辑与相应物理设备之间的映射关系，隔离各个虚拟网络的转发表以及流量，其核心是把网络中所有的流量分类到多个“分片”，将每一个“分片”交由一个 SDN 控制器进行控制。

6.4 基于 SDN 的异构无线网络接入选择策略

异构网络融合的一个主要内容就是让用户可以在不同网络中自由地接入和切换，给用户一个无缝的服务体验。在接下来的两节中，将介绍基于 SDN 的异构无线网络接入选择策略和基于 SDN 的异构开放无线网络无缝切换技术。

在异构开放无线网络架构中，利用软件定义无线电(SDR)技术对传统基站进行改进，目前基本实现 LTE 软基站、WLAN 软接入点(AP)和 GSM 基站等设备的可编程。数据层通过南向协议与控制层相连。控制层由控制器组成，控制器是整个网络架构的核心，具有网络状态感知、QoE 感知、设备行为管理、移动性管理和无线资源管理等功能。

整体异构无线网络接入选择架构如图 6-11 所示。

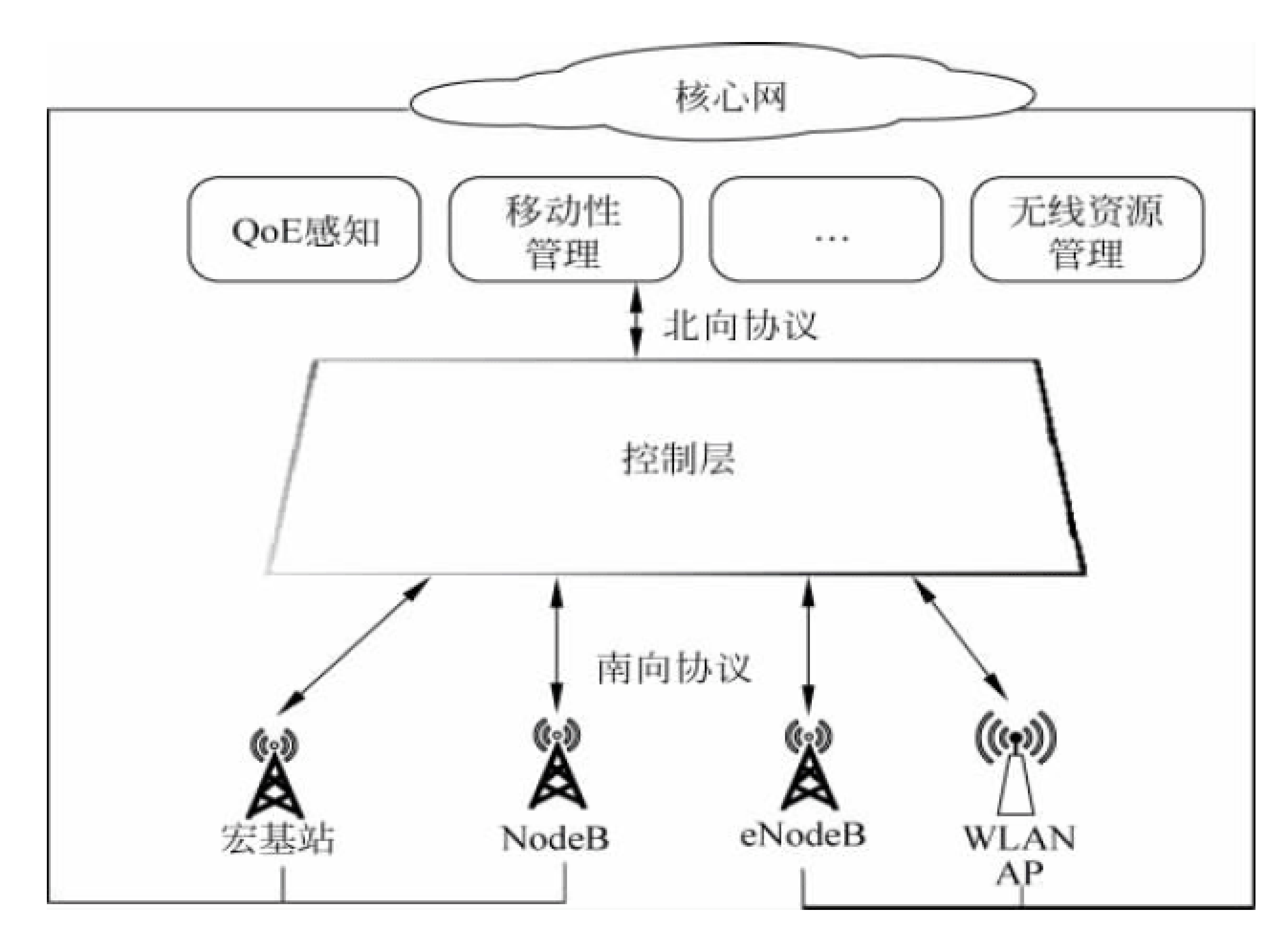


图 6-11 基于 SDN 的异构无线网络接入选择架构

控制器的主要功能包括：

(1) 系统配置——负责系统的基本信息配置，并与数据层进行交互。

(2) 移动性管理——管理移动终端在异构无线接入网络之间的切换等操作，并执行接入控制等功能。

(3) QoE 感知管理——感知不同业务的 QoE，并负责业务流的触发、维持、转变和消除等工作。

(4) 无线资源管理——进行业务流量的调度并执行基于 QoE 的资源分配和信道质量管理。

(5) 物理层的管控——提供接口以实现信道分配和切换工作，也提供接口来执行链路自适应技术，主要包括信道质量感知的评估和反馈、调制方式和编码策略的调整以及信号功率控制等。

为了实现控制器的集中控制功能，采用逻辑上集中、物理上离散的控制器架构。控制器由接入网控制器(局域控制器)和全局控制器两部分组成，其中接入网控制器作为决策制定者，为数据流制定自定义的转发方式，全局控制器负责各接入网控制器之间的通信。所有的控制器都具有同等级别的网络级的控制能力，它们运行相同的控制器软件和应用设置，每个基站或接入点都连接到处于通信范围内最优的接入网控制器上，每个控制器控制与其直接相连的基站或接入点，通过全局控制器实现与其他接入网控制器之间的通信，从而能够间接地询问和管理其他的基站和接入点，一旦某个控制器出现故障，与其相连的基站和接入点可以通过特定的接口进行重配置，并与其他处于通信范围内的接入网控制器建立连接。为了实现全局的控制能力，接入网控制器会周期性地进行数据交换，更新系统状态。控制器部署架构模型如图 6-12 所示[7]。

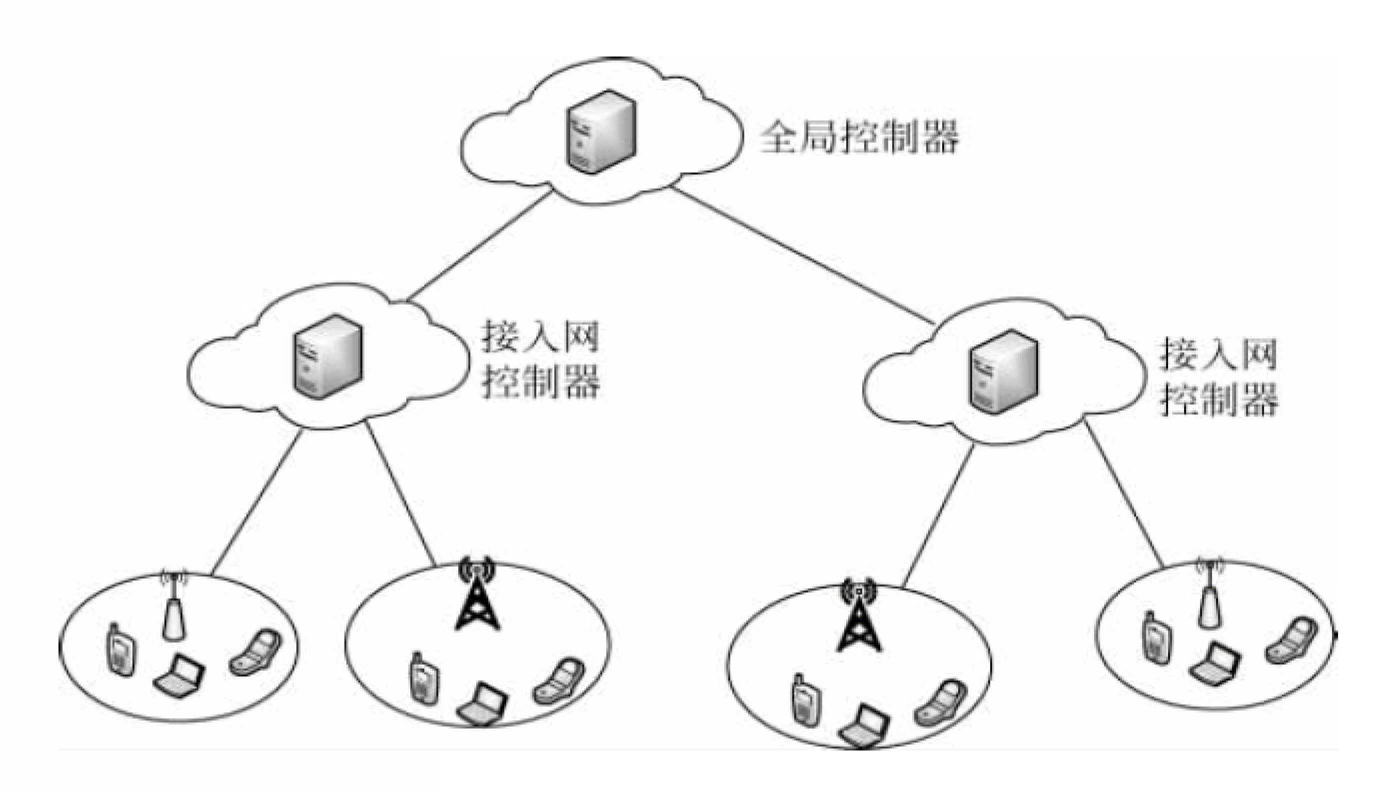


图 6-12　控制器部署架构模型

6.5　基于 SDN 的异构无线网络无缝切换技术

切换作为移动通信系统最重要的功能之一，它的性能好坏与服务质量及用户的通信体验息息相关。切换主要分为两种，一种是发生在同种网络之间的水平切换，另一种是发生在异构网络之间的垂直切换，而且垂直切换要更为复杂。

现阶段对垂直切换技术的研究仍然面临着诸多挑战。首先，由于切换时需要知道其余网络的信息，因此切换判决需要在上层来实现，这会在一定程度上降低切换性能。其次，现有的切换算法存在着可行性差、乒乓切换等问题，设计一个高效可行的异构网络切换算法是当前研究的重点。除此之外，由于异构网络环境的复杂性，如网络条件、终端状态以及用户偏好等，每一个因素都对网络切换起着影响，因此，切换判决需要综合考虑众多因素。

异构网络间的垂直切换一般分为三个步骤：最佳网络选择，执行切换判决，切换信令交互。在整个切换过程中，不同的参数会在不同的过程中起着或大或小的影响，如资费、带宽等相对稳定的用户偏好参数主要影响最佳网络选择，而终端的运动速度、所在位置以及接入点的信号强度等时间短变化快的参数主要影响切换判决。在切换判决过程中，如果不能很好地调整各参数的权重，就可能会引起乒乓切换、连接中断等问题[9]。

由于切换是中断与某个网络的连接并与另一个网络建立连接，是正在进行的业务最容易发生中断的时刻，因此如何在切换时保证业务的连续性，降低切换对用户的影响，甚至让用户感受不到切换的发生，是实现无缝切换的关键。

为了保证业务的连续性，本节将介绍两种新型异构无线网络垂直切换机制：IP 地址再封装法和源路由再封装法[7]。在控制器做出切换判断后，控制器将被切换业务在原网络中的配置信息及资源占用信息告知新网络，由新网络为即将切换来的业务提前分配好网络资源，降低了切换的时延。

1. IP 地址再封装法

具体的切换场景如图 6-13 所示。某个用户将从 LTE 网络切换至 WLAN 网络，切换

前移动终端的业务在 LTE 网络中的本地地址是 ip1，对应的响应节点的 IP 地址为 D。当终端切换到 WLAN 目标网络后，会被分配一个新的本地地址 ip2。为了保证切换发生后在 WLAN 网络中当前会话的连续性，在上行方向，将当前会话中以 ip1 为源地址的 IP 数据包进行重新封装，并以 ip2 作为源地址，目的地址 D 不变，利用 IP 网络传输数据包，因此该数据包能够到达响应节点。响应节点收到该数据包后，先进行解封装，将提取出的原始数据包发送到内核中的 IP 栈内，响应节点仍会把这个原始数据包当作是从原始 LTE 网络接收而来的，从而保证了上行会话的连续。而在下行方向，当响应节点 D 发送数据包到终端时，会话会使用 ip1 作为目的地址，这些数据包通过 LTE 网络发送到终端。总体来说，IP 地址再封装法能够实现上行方向的会话连续，但在下行方向并不能实现会话连续，仍有可能发生丢包[7]。

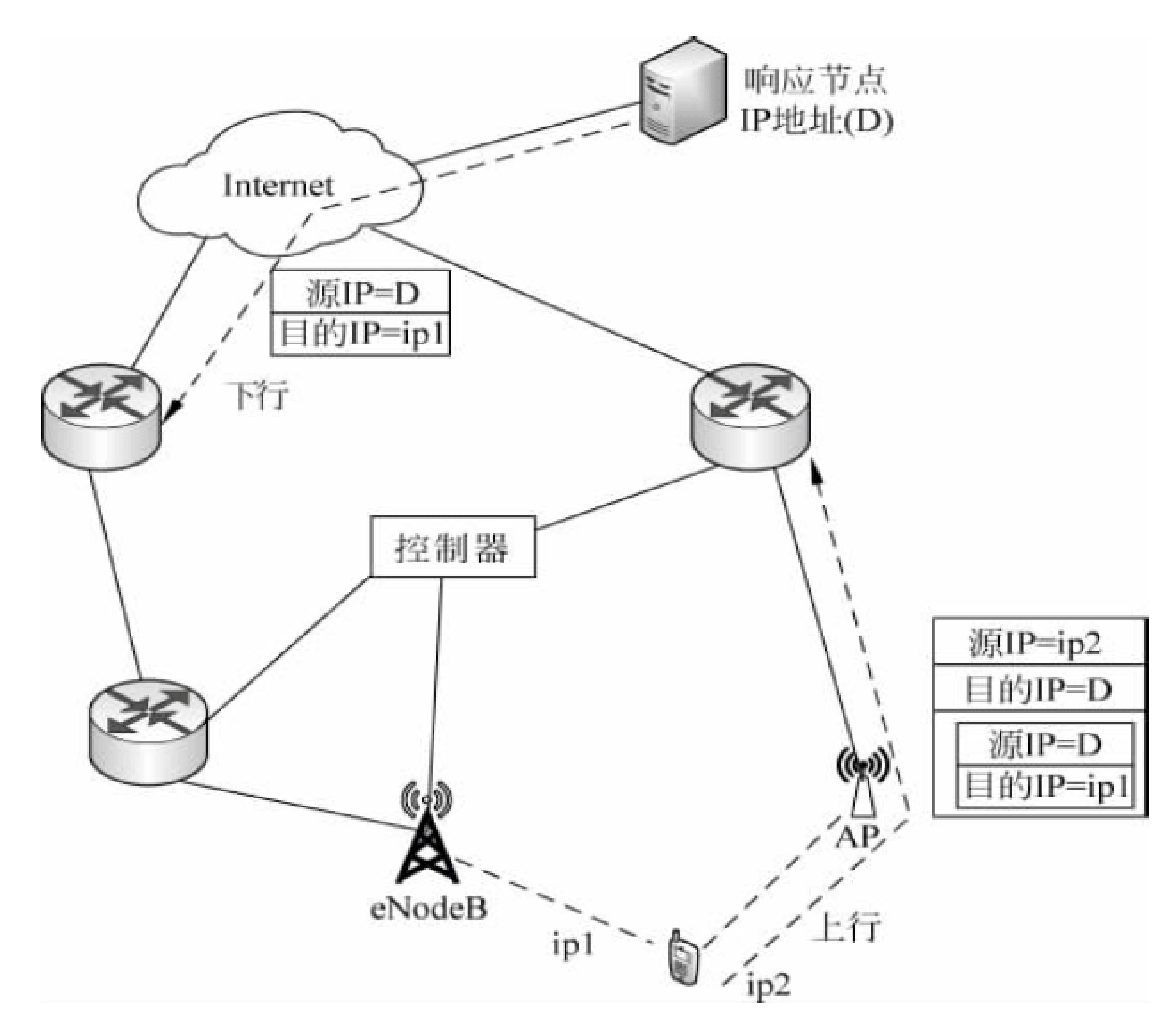


图 6-13　IP 地址再封装法

2. 源路由再封装法

由于 IP 地址再封装法不能解决下行方向的数据包丢失问题，因此结合源路由技术，进一步提出了源路由再封装法。如图 6-14 所示是具体的切换场景图。在由 LTE 网络向 WLAN 网络切换时，终端产生的 IP 地址为 ip1 的原始数据包会将源路由信息附加到该数据包中。源路由信息指明了源数据包的转发过程。移动终端将包含源路由信息的数据包封装到源地址为 ip2、目的地址为 D 的数据包中。通过再封装过程，数据包能够在 WLAN 网络中传输。响应节点 D 由 WLAN 网络收到该数据包之后，通过解压缩从中得到源数据包，从而在源数据包中进一步获得源路由信息。响应节点通过使用逆向的源路由进行目的地址是 ip1 的下行数据包传输，这些下行数据包经过 WLAN 网络发送到 ip2。移动终端在内部处理源路由信息的过程会将 IP 数据包由 ip2 转发到 ip1 中。因此，源路由再封装法能够实现在下行与上行方向的会话连续性。对于网络不支持源路由法的情

形，移动终端将不能使用源路由封装法，作为替代，可以在移动终端中应用端到端的双向隧道。当终端触发了和响应节点之间的双向隧道时，移动终端在 LTE 网络中所发送的数据包{ip1,D}将包含包头{ ip2,D}和隧道包头。响应节点发送的数据包{D,ip1}也需包含外部包头{D,ip2}和隧道包头[7]。

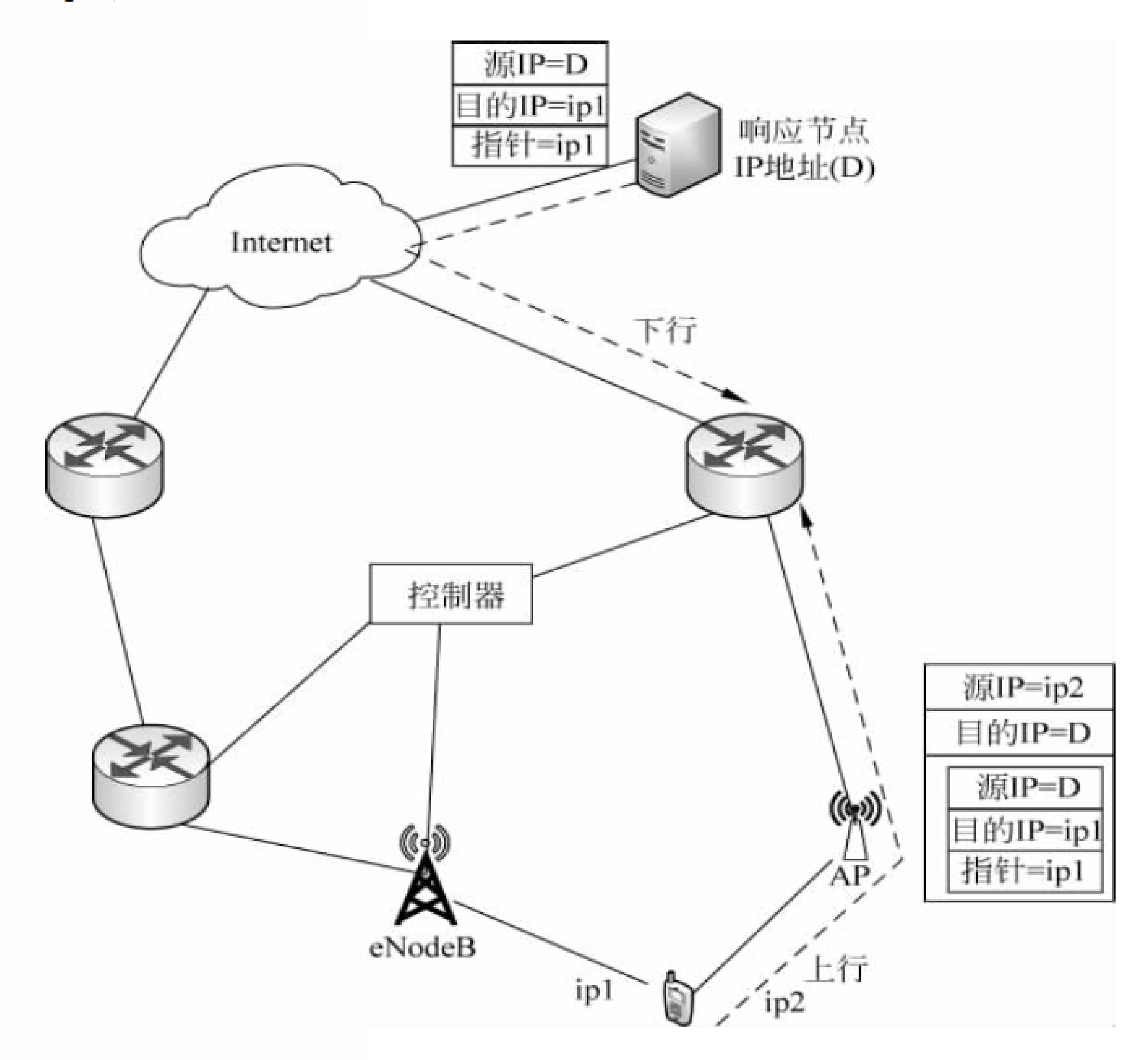


图 6-14 源路由再封装法

6.6 基于 SDN 的异构网络绿色节能方案

能源主要分为两类：一类是以煤炭、石油、天然气等为代表的化石能源，另一类是以水能、风能、太阳能等为代表的可再生清洁能源。由于化石能源所带来的资源紧张、气候变化、环境污染等问题日益突出，低碳清洁能源的使用已成为当前能源发展的必然趋势。异构网络节能问题也逐渐被人们重视。

6.6.1 异构网络能耗问题研究

近几年来，随着互联网技术的迅速发展，网络业务量呈现急剧增长的趋势，思科未来网络发展业务趋势白皮书的统计数据[10]显示，截至 2014 年，全球 IP 流量已经达到 59.9 EB/月，预计到 2019 年将增长至约 3 倍，达到 168.4EB/月。为了满足骤增的网络业务需求，需要更多高性能与高可靠的网络设备，这些设备除了自身能耗，还需要依赖制冷机制与网络基础架构[11]，因此造成的高碳排放量使得信息和通信技术产业(ICT)的能耗问题需要得到广泛的关注。根据欧盟发布的报告[12]，如果在 2020 年之前将全球升高的气温

控制在 2℃以下，温室气体的排放体积需要减少 15%～30%。而计算机和网络设备的能源消耗正在成为一个全球能源消耗的重要组成部分[13]。根据 2009 年的统计数据[14]，ICT 产业消耗的电力占据全球总耗电量的 8%。除了对生态环境产生的影响，碳排放量同时限制了 ICT 产业的经济效益，从可持续发展的角度来看，未来网络必然以绿色节能为发展趋势，需要从网络架构、网络管理、网络承载等各个层面以能效优化与增强绿色能源利用率为目标对当前网络进行综合改革和优化。

另外，随着网络服务类型的不断增加，用户的需求也千变万化。传统单一网络类型已不足以满足用户需求，多种类型网络共同为用户提供服务的异构网络应运而生。随着新的网络接入技术的不断出现，用户可接入的网络种类越来越多，同时当今网络的异构性不仅存在于实体资源之中，还存在于业务类型之中。然而随着网络业务量增加，异构网络的弊端逐渐显现。在网络结构上，由于现有网络采用垂直建站的组网和运营模式，使得异构网络之间无法实现有效互通，这就造成了管理上的混乱，难以从全局角度对资源进行优化调度，不仅没有给用户提供更高效便利的网络服务，还造成大量设备以及资源的浪费。从管理模式上，现有的网络中控制和转发紧密耦合，因此在现有网络上改变拓扑结构，需要分别对软件和硬件进行配置，存在布线困难、消耗大量人力和物力资源的问题[15]。Nick 等通过软件定义网元设备，通过软件的形式改变网络的拓扑结构，并延伸出一个新的网络技术——软件定义网络（SDN）[16]。SDN 作为一种新型的网络范例[20]，打破了传统网络的垂直组网模式，将网络控制逻辑从基础层的路由器和交换机中分离出来，实现了可编程网络，更加高效地路由和更加高效地使用链路资源。同时这样的网络更容易创建并引入网络抽象，简化了网络管理并促进网络演进。

在数据中心网络中，可以利用 SDN 通过全局网络信息消除数据传输冗余[17]，从而避免了网络资源的浪费。参考文献[18]将 SDN 应用在了数据中心路由中，利用 SDN 掌握全网信息，通过对流按时间进行调度，让每条流在某一时间段独占带宽，减小了流的竞争时间，使链路得到充分利用，从而节省了数据中心的能量。而参考文献[19]利用了 SDN 掌握全局信息的能力，实时关闭暂未使用的设备，当有需要时再打开，节省了约一半的能耗。

在传统异构网络中引入可编程网络理念，一方面可以从全局的角度出发，根据应用层不同业务需求（如带宽、时延需求等），结合底层实体资源的状态（负载量、用户数、剩余绿色能源、频谱资源等），针对特定的目标，来对网络设备进行相应的策略调整。另一方面由于控制器的存在，转发设备可以不再维护路由表，而是按照控制器下发的流表规则进行转发，同时转发设备实时地向控制器发送自己的设备信息（如各转发设备的剩余绿色能源量、预测绿色能源量、实时负载量以及可用频谱资源等参数），控制器收到这些信息后存储在数据库中，并构造全网能源资源拓扑图，控制器在综合考虑用户的 QoS 需求及底层设备信息之后，做出最合适的路由选择。最后，运营商为了解决网元过多问题而提出的 SDN 方案——网络功能虚拟化（NFV）技术[21]，将网络转发设备向可编程化、标准化推进，进一步减少设备层面上的资源浪费，因此，对于可编程网络的研究，在节能减排与提高资源利用率方面具有巨大的应用前景。

如图 6-15 所示，在异构网络场景下引入软件定义网络思想，以用户 QoS 保障为基础，以最大化网络资源利用率并减少网络能耗为优化目标，对传统异构网络架构进行改

进，实现了灵活的网络管理方案、高能效的网络资源调度策略与更好的用户服务体验。

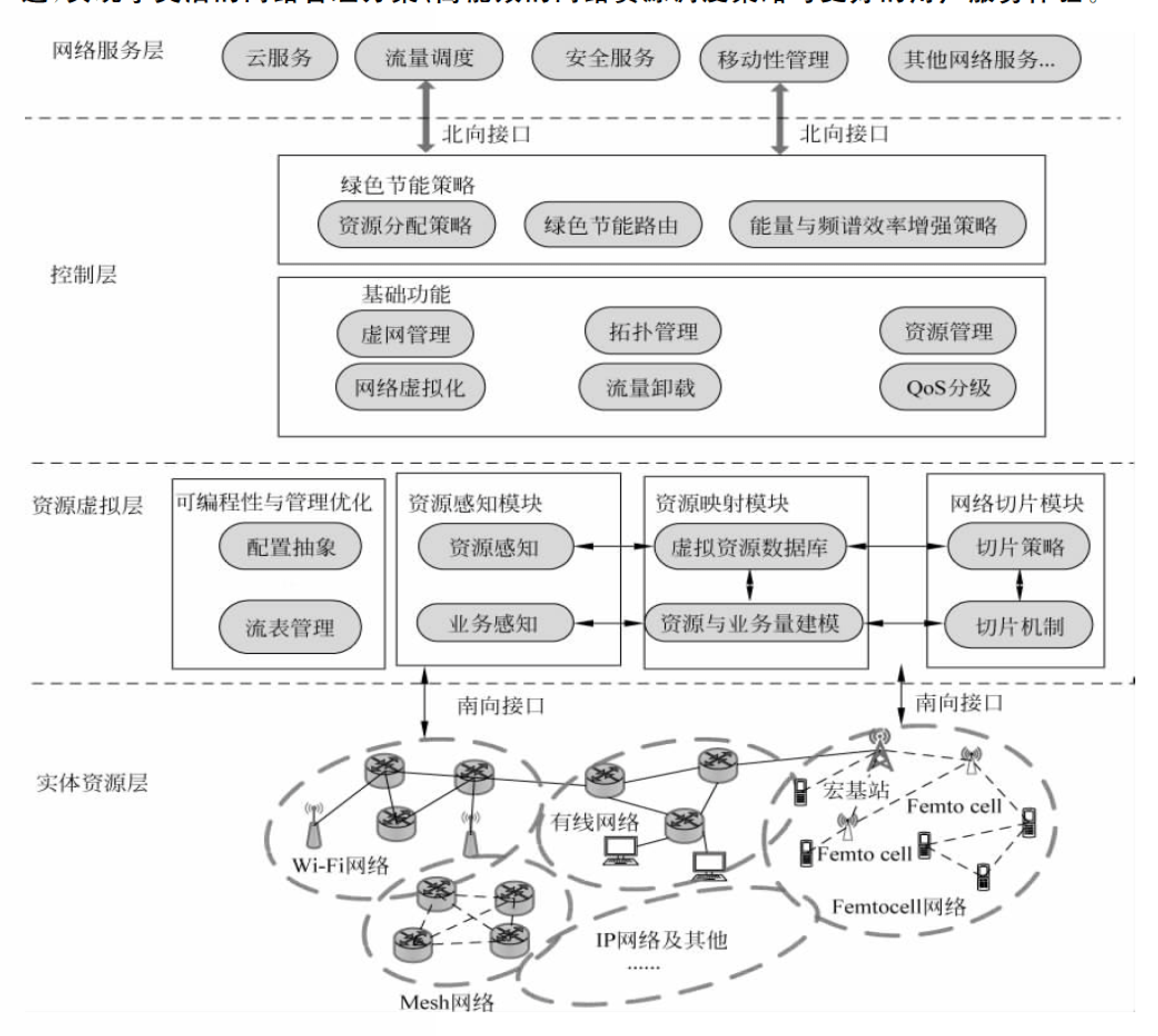


图 6-15　基于 SDN 的异构网络绿色节能方案整体框架图

6.6.2　异构网络节能方法简介

异构网络由于密集型网络的部署，网络设备能耗巨大，近年来异构网络下的节能技术得到了大量的研究。目前，主要在以下几个方面进行异构网络节能技术研究。

（1）先进的物理层技术。如 OFDMA、载波聚合、高阶 MIMO、阵列 MIMO、认知无线网络和网络编码等，各项先进物理层技术研究中，节能都是重要的考虑因素之一。

（2）网络架构。随着异构网络的发展和变化，网络架构也得到了越来越多的研究，如 SDN、移动云计算以及融合大数据的 SON（self-organized network，自组织网络）等，都能提高网络资源利用率，从而降低网络能耗。

（3）无线资源管理。由于异构网络中存在多接入技术，不同的技术对资源的使用和

性能也各不相同，因此需要进行异构网络无线资源管理来进行资源分配，提高资源利用率。而在同种接入技术网络下，无线资源管理主要是频谱分配和功率控制。通过合理的资源分配算法，不但能够降低全网功耗，还能提高用户的服务体验。

（4）网络切换策略。在异构网络中，由于网络状态的变化及用户行为的变化（如用户移动、用户设备电量变化等），可能会导致用户当前接入的网络不再适合为用户服务，需要切换至其他网络。根据网络接入技术的相同与否，切换又分为水平切换和垂直切换。水平切换前后用户所接入的网络都属于同种网络（属于同一通信系统），如 LTE 中的宏基站和微基站；而垂直切换前后用户所接入的网络属于不同的网络，如 Wi-Fi 和 LTE。利用有效的网络切换策略，不但能够提升网络节能性能，而且能保证用户的服务体验。

（5）基站睡眠策略。异构网络由多种不同发射功率的基站组成，由于用户行为的潮汐特性及区域之间的差异，往往导致区域性负载不均衡，有的基站满载甚至超负荷工作，而有的基站却很少有用户接入，甚至是无服务用户，因此，可以在满足网络负载均衡的情况下对基站进行休眠。基站休眠策略中，主要的休眠对象是低功率基站，因为高发射功率基站覆盖的区域较广，相比于低功率基站，负载要大很多。当微基站、毫微基站以及家庭基站等低功率基站在负载低于门限值时，进入休眠模式，让高发射功率基站（如宏基站）来为处于休眠基站覆盖范围内的用户服务，在保证用户服务的同时，通过低发射功率基站的休眠来节省全网能耗。

（6）小区协作。异构网络中包含了多个小区，为了优化整体网络性能，最大限度节省网络能耗，需要进行多小区协作，交换小区间信息，完成系统级的优化。小区协作包括多个方面，如小区间干扰协调、小区间负载均衡。此外，设备定位等高级应用也需要小区间的合作。

（7）基站部署策略。宏基站的部署需要经过非常精细的理论分析和实际测量，成本较高。对于业务盲区及业务忙区，通过建立微基站作为宏基站的补充，如果基站部署不合理，会导致资源利用率低、能量浪费。目前异构网络中基站拓扑控制采用混合模式，对于宏基站采用同构网络中的精密部署策略，而对于低发射功率基站灵活部署，节省了基建成本。此外采用有效的休眠控制方式，通过基站间协同来提高网络资源利用率，降低网络能耗。

6.6.3 基于 SDN 的节能技术简介

利用 SDN 的集中管控、灵活编程等特点，可以很好地执行流量管理、资源分配、路由决策等，进一步促进了网络节能技术的实现。

与传统网络相比，SDN 控制器可以对整个网络系统全局管控，它的可编程性赋予其强大的软件功能。因此，从软件角度考虑，目前的 SDN 节能策略从资源分配、流量管理等方面着手，主要是将一些低利用率或闲置的网络单元（如交换机、路由器、链路）转入睡眠状态，使用尽可能少的路径传输全网数据流。另外，因为服务器是耗能的关键，关闭低利用率或不必要的服务器也是节能的重要手段，主要方法是把低利用率服务器上的虚拟机合并到其他服务器中。再从硬件角度考虑，SDN 交换机的存储硬件 TCAM（ternary content addressable memory，三态内容寻址存储器，主要用于快速查找 ACL、路由等表

项)是个极端耗能的设备,提高其利用率可以较好地降低能耗。通过对流规则进行优化,压缩查找转发规则,优化利用了 TCAM 中的存储空间。这里将基于 SDN 的节能技术分为三类:睡眠策略、流规则布局优化技术和虚拟机合并策略[22]。

1. 睡眠策略

在整个网络中,经常有一些网络组件处于低利用状态或空闲状态,因此可以将一些没有使用或处于较低利用状态的资源(如交换机、端口、CPU 等)关闭或转换为睡眠状态。这种方法可以结合流的动态管理实现,如根据网路流量负载,合并现有流或选择最佳的路径等。利用 SDN 的集中、灵活的资源管理功能,可以实现流的动态路由转发。除了将网络组件关闭或使其处于睡眠状态,还可以根据链路利用率调整链路的传输速率,即在链路低利用率状态下,降低链路的传输速率,达到节能的目的。

(1) 优先级流调度。利用 SDN 技术,可以动态管理流的转发路径。考虑不同应用的 QoS 需求和经过交换机的能量消耗,赋予不同数据流不同的优先级。高优先级的流优先传输,这样可以选择现有的活跃交换机进行数据传输,最大限度地不去激活那些处于休眠状态的交换机,达到降低功耗的目的。

(2) 相关性感知的能量优化算法。相关性感知能量优化算法的设计依据是:一是从数据流的轨迹来看,不同的流量的相互独立,即不同流的带宽需求不会在同一时刻到达顶峰;二是对于大多数链路,大概 90%的时间链路使用率都远远低于峰值。因此如果在流合并过程中,考虑到流量路径之间的相关性,可以占用更少的链路,提高链路利用率,节省更多的能量。另外,把流汇聚和链路速率自适应结合,可以进一步提高节能效果。对于流汇聚和链路的数据传输速率问题,利用线性编程工具可以达到一种近似最佳的解决办法。

(3) 链路速率自适应。网络组件在设计时往往会以满负载标准时工作,而实际情况中链路在大部分时间处于低利用率状态。链路数据传输速率越大,能量消耗越大。因此,降低低利用率链路的数据传输速率可以有效地减少能量消耗。

2. 流规则布局优化技术

CAM (content addressable memory)是内容寻址存储器,写入 CAM 的数据会和其内部存储的每一个数据进行比较,并返回与端口数据相同的所有内部数据的地址。TCAM 是一种特殊的 CAM,可以视为一组固定宽度的流表项。每个 TCAM 流表项由三元数字构成,即 0、1 和 x(任意值)。对于输入 x,x 代表通配字节,可用来基于模式匹配进行更广泛的搜索。由于 TCAM 的快速查找性能,其在 SDN 交换机中得到广泛应用,但其价格昂贵且十分耗能。

TCAM 的能耗过多主要由两个因素引起:一是硬件本身的耗能,1MB 的 TCAM 芯片耗能 15～30 W,SDN 网络有更好的细粒度,其流表项更长,占用空间更大;二是由于 TCAM 的资源限制,使得存储在 TCAM 中的流表项十分有限,导致没有足够的流规则能更好地转发数据流,使得节能效果不明显。

针对上述两个问题,可以考虑从两方面解决。流表项一般由头域、计数器和操作项组成。每个流表项就是一个转发规则。如果在满足功能的情况下,能用更短的标签代替

现在的流表项组成方式，流表项所占的字节数就变小了，可以实现 TCAM 能耗的降低。另外，即使不改变流表项本身所占空间的大小，但是可以提高流规则的利用率。换句话说，就是在满足功能的情况下，所需要的流规则数量减少了，TCAM 的利用率提高了，也能实现能耗的降低[22]。

3. 虚拟机合并策略

研究表明，典型情况下，许多服务器仅以满负载状态的 30%运行。通过虚拟化技术将一台计算机分为多台逻辑计算机，每台逻辑计算机可以独立运行不同操作系统，各操作系统中的应用程序相互独立而互不影响。因此，提高服务器的利用率，利用更少的服务器创建更多的虚拟机，可以降低硬件的能量消耗。虚拟机合并是解决此问题的关键技术之一。通过虚拟机合并技术，关闭了一些低利用率的物理服务器，可以降低能耗。管理员通过云控制器提供的接口对虚拟机进行配置，并对虚拟网络进行设置。SDN 控制器利用 OpenFlow 协议实现对网络设备的管理，同时 SDN 控制器通过与云控制器的交互，实现网络检测、QoS 支持、路径计算、安全设备管理的功能。云控制器根据用户需求为虚拟机分配资源（CPU、存储资源、带宽），并根据需求的变化，在虚拟机合并后，选择开启或关闭服务器。

6.6.4 SDN 在能源互联网中的应用

能源网、交通网、信息网是全球最重要的三大基础网络。全球交通网、信息网已经总体建成、实现互联，能源网必然向互联方向发展。

美国著名学者杰里米·里夫金认为，由于化石燃料逐渐枯竭，加上它造成环境污染问题，以大规模利用化石燃料为特征的工业模式正在走向终结。以新能源技术和信息技术的深入结合为特征的一种新的能源利用体系即"能源互联网"（energy Internet，EI）即将出现，而以能源互联网为核心的第三次工业革命将给人类社会的经济发展模式与生活方式带来深远影响[23]。他在《第三次工业革命》中提出了能源互联网的愿景，但并没有给出能源互联网明确而严格的定义。

2015 年 9 月 26 日，习近平主席在联合国发展峰会上宣布，"中国倡议探讨构建全球能源互联网，推动以清洁和绿色方式满足全球电力需求"[24]，为世界能源绿色低碳发展描绘了新蓝图，为应对气候变化开辟了新道路，向世界彰显了中国智慧和中国担当。在 2016 年第一季度，国家发改委、能源局和工信部联合发布《关于推进"互联网＋"智慧能源发展的指导意见》[25]，明确未来能源互联网的发展路线和思路及中远期十大重点任务，下一步能源局则将组织试点项目，先试点后推广，分两阶段推进，实现能源的互联互通。构建服务范围广、配置能力强、安全可靠性高、绿色低碳的全球能源互联网是推动全球清洁能源高效开发、配置、利用，实现世界能源可持续发展的必由之路。在能源和电力需求增长的驱动下，世界电网经历了从传统电网到现代电网，从孤立城市电网到跨区、跨国的大型互联网的跨越式发展，并进入了以坚强智能电网为标志的新阶段。构建全球能源互联网，可为全世界乃至整个人类社会发展提供更安全、经济、清洁、可持续的能源。

能源互联网是以互联网理念构建的新型信息网——能源融合"广域网"，它以大电网

为"主干网",以再生式能源等能量自治单元为"局域网",以开放对等的信息网——能源一体化架构实现真正的能源双向按需传输和能源动态平衡使用[24]。能源互联网主要表现为由可再生清洁能源转化为电能而带来的电能流动。

1. 能源互联网架构

全球能源互联网是以特高压电网为骨干网架、全球互联的坚强智能电网,关键在于构造一种能源体系,使得能源像互联网中的信息一样,可以以开放、互联、对等、分享的方式真正实现能源的双向按需传送和动态平衡使用。

能源互联网是在现有电网基础上通过先进的电力电子技术和信息技术融合大量可再生清洁能源发电装置和分布式储能装置,实现能源和信息流动的新型高效电网结构,是升级版的智能电网。

其中,可再生清洁能源的开发分为集中式开发和分布式开发两种情况。集中式开发一般远离负载中心,人口稀少,平均日照强或年均风速高,单位面积资源量大、可开发时间长,一般通过大规模电网等方式实现电力输送;分布式开发的可再生清洁能源一般分布在负载中心附近,人口集中,通常阳光和风速温和,可以满足一部分用电需求,对采集的能源使用就地利用的方式,如果有剩余的电能就存储下来,如城市屋顶光伏、生物质能等。

通过局域自治消纳和广域对等互联,能源互联网最大程度地适应了可再生能源的动态性。此外,能源互联网在安全、可靠、稳定以及利用率等方面优势明显。相比于集中式电网自上而下的紧耦合模式,能源互联网在广域互联中可通过储能缓冲、直流输电等方式实现解耦,同时通过广泛互联实现广域的动态互备用,解决了局域能量不稳定问题,实现了能源的安全稳定可靠供应。

能源互联网不是取代现有的电网架构,而是借鉴互联网理念提供一种自下而上的新型组网方式,在分布式可再生能源越来越受人们重视、混合能源供能场景越来越普遍的形势下,能源互联网通过局域能量自治和广域能量交换最大限度地削弱了分布式能源动态随机的缺点,大大提高了大电网的安全稳定性,是对现有大电网架构的有益补充。

在我国,电网为紧耦合架构,传统电网模型把主干电网的电源看作无穷大,微电网或分布式能源的电量并不会对主干电网产生太大影响。然而,当可再生清洁能源与配网紧密渗透,由此带来的电压上升、频率调整能力不足、供电不稳定等问题将会对主干电网的监控和管理带来一系列的冲击和考验。我国电网的管理模式是自上而下的垂直管理模式,而微电网和分布式能源灵活多变,因而在能源互联网的管理和建设模式上应采用以集中模式为主,分散自治模式为辅的方式。传统电网一般根据电力系统的历史发电量来调控负载并决定调度方式,通过分析计算电力系统安全稳定条件,靠经验来预测未来用电负载变化。能源互联网要求发电量随着负载的变化而变化,保持发电量相对稳定,以节省发电费用,减少二氧化碳排放。在能源互联网中,传统电网的"离线预测、实时匹配"的控制方式将向着"实时计算、实时控制"的模式发展。

国内以曹军威为代表的学者提出了能源路由器的概念并给出了能源路由器的实现方式[27]。能源路由器可分为能源互联网间路由器(下称"局域网能源路由器")和能源互联单元与主干网的并网路由器(下称"主网能源路由器")。能源路由器可分为网络层和

能量层，网络层含有能源信息采集模块，能量层通过电子电力变压器实现变压调节、交直流转换、能量的存储及稳定控制，还实现微电网或线路开关的智能通断控制。网络层提供相应的通信接口，如电力线通信、以太网、Wi-Fi、ZigBee 接口。在能源互联网中，主网能源路由器能够实现 220 kV/110 kV/35 kV/10 kV 的变压调节、交流和直流的相互转换，局域网能源路由器则具备储存和使用可再生能源的能力，并尽可能高效地利用可再生能源。在能源互联网中，能源路由器还通过感知能源流的实时流动，控制能源数量和流动方向，进行负载监控和计量，同时将能源的流动信息传送到相关调控中心。

2. 能源互联网信息通信架构

与能源电力相对完备的基础设施相比，能源互联网中信息通信领域的基础设施目前尚处于起步阶段，而能源互联网的分散协同调度与控制更需要实时信息采集、传输、在线分析与决策。为更好地协调用户和电网之间的供需关系，电力系统需要实时采集和处理海量数据，分析用户的用电习惯，实现实时电价查询和用户个性化的用电选择，并为调控中心的“在线预测、决策，实时匹配”提供决策依据，而海量数据的存储和处理则促使调控中心和营销中心相关主站服务器向具备云计算、云存储功能的云端服务器转换[28]。目前，电力系统中的信息系统和通信网络彼此是独立的。因此，为了保证物联网、智能移动终端、传统信息通信网络交互融合的网络架构能够适应风能、太阳能、海洋能等间歇式可再生能源的友好接入，迫切需要支持信息通信网络交互融合的网络通信技术的支撑。

依据智能电网现有的三级调控架构，能源互联网在建设和管理模式上采用以集中模式为主，以分散、合作自治模式为辅的方式，网络组网及物理承载方式则可利用现有智能通信网络资源，并在能源路由器上集成多种通信接口，如 TD-LTE 4G/5G 接口、光纤网络的 EPON/GPON 接口及 SDN 接口。SDN 网络能够通过应用程序接口感知用户的动态业务需求，如带宽、时延、地理位置等，通过 SDN 控制器可以根据这些业务信息来提供更好的服务响应并提高网络资源利用率。

参考文献[29]提出了一种信息通信架构。该架构由 IEC(intelligent energy controller，智能电网控制器)、数据中心和 SDN 控制器组成。在该体系中，智能电网控制器能够实时采集能源信息并进行预处理，然后将数据传送到数据中心，数据中心再次对数据进行处理并返回给智能电网控制器。智能电网控制器根据这些结果动态调整发电量和能源存储量。智能电网控制器通过现有接入网实现信息互通：在数据中心处安装 SDN 路由器和交换机，利用控制器管理信息流，进行流量控制和负载均衡。一个单一的 SDN 控制器不仅控制它自身管辖范围内的数据中心，同时还在邻近区域的其他数据中心充当备份控制器。即每个数据中心至少连接两个 SDN 控制器，但每个数据中心在同一时间只能受一个控制器控制。

结合现有的分布式能源的实施政策，地区局域能源互联网通过接入网实现可再生能源微电网与地市/县级调控中心之间的数据通信，并实时采集电压、电流、有功、无功、发电量等运行状态；通过家庭智能电表等智能交互终端采集用户的用电习惯及行为偏好并上传至数据中心进行数据分析；通过信息共享，更好地感知电网可用电量和区域负载状况，更准确地匹配电源与负载，消纳可再生能源，提高全网整体能效，避免能源系统的过度冗余配置。局域能源互联网控制器可设置在地区调控中心；跨区域能源互联网可将节

点设置在跨区域调控中心，利用汇聚算法对能量单元信息进行汇聚处理和流量工程管控，减少冗余数据的开销，提高通信效率。

基于 SDN 网络架构，可以将分散的网络硬件单独配置管理模式转换为集中管理模式，在电力调控中心设置统一的监测管理系统，从而实现网络系统的集中配置、统一监管，降低了系统配置难度，减少了维护及通信开销[30]。

6.7 基于 SDN 的网络设备部署方案

网络部署有两种形式。一种是 Underlay，这种网络中，所有的转发行为都由控制器决定，控制器通过 OpenFlow 协议或者定制的 BGP 协议将转发表项下发给转发器，转发设备根据转发表进行动作，没有单独的控制面。另一种是 Overlay，这种网络的转发设备一般由传统设备组成，不支持 OpenFlow，或者不能部署私有定制协议。这时就要用到隧道技术[隧道技术(tunneling)是一种通过使用互联网络的基础设施在网络之间传递数据的方式。隧道协议将其他协议的数据帧或数据包重新封装然后通过隧道发送。新的帧头提供路由信息，以便通过互联网传递被封装的负载数据]，以传统网络为基础，通过路由协议实现设备互通，在服务器接入点，采用隧道技术将数据报文进行封装或者解封装。虽然 Overlay 部署方案能够在不对现有网络设备进行大范围改造的前提下增加新的功能，但无疑会降低网络性能，因此，随着设备的淘汰与演进，Underlay 方式应该会是今后 SDN 网络的趋势。由于当前专用 SDN 设备还未实现大规模部署，因此 Overlay 网络目前是 SDN 网络部署的主流方案。本节中将重点介绍 Overlay 组网方案。

覆盖网络(overlay network)是在现有网络基础之上构建的一个虚拟网络。在覆盖网络中，节点可以是网络中各种功能节点(路由器、服务器、用户终端等)，节点之间的虚拟链路是逻辑上的定义，通常对应于底层网络的物理路径。目前，覆盖网络技术得到了广泛的应用，主要有基于文件共享和实时流媒体服务的 P2P(peer-to-peer)覆盖网络、内容分发网络(content deliver network，CDN)、应用层组播(application layer multicast，ALM)等。随着覆盖网络技术的研究深入，覆盖网络除了能够提供各种新型的应用之外，研究学者开始使用覆盖网络技术提高 Internet 的服务性能，如传输速率、QoS、可靠性等。

6.7.1 Overlay 技术产生背景

1. 产生背景

随着企业业务的快速扩展，对网络基础设施提出了更高的要求，企业需求的重点也转变为新业务的快速部署和提高设备利用率。由于云计算可以方便地为企业按需提供资源，已越来越受到当前企业 IT 建设的欢迎，而作为云计算的关键技术，虚拟化也越发受到重视。随着虚拟机需求的快速增长，以及虚拟机迁移成为一个常态性业务，传统网络模式已经不能满足企业需求，面临着如下挑战[31]。

1) 虚拟机规模受限

在二层网络中，数据流均需要通过明确的网络寻址才能准确地到达目的地址，因此

网络设备二层地址表项(即 MAC 地址表)的大小,决定了云计算环境下虚拟机规模的上限,并且由于表项的有效性并不能达到百分之百,使得可用的虚拟机数量进一步降低。特别是对于低成本的接入设备而言,因其表项规格较小,限制了整个云计算数据中心的虚拟机数量。

2) 虚拟机迁移范围受限

当虚拟机从一个物理设备上迁移到另一个物理设备上时,为了不间断虚拟机业务,需要保持 IP 地址、MAC 地址等参数不变,因此要求业务网络是一个二层网络,同时网络本身具有多路径多链路的冗余和可靠性。传统的网络生成树(spaning tree protocol, STP)技术不仅部署烦琐,而且协议复杂,不能适用于大规模网络,限制了虚拟化的网络扩展性。基于各厂家私有的 IRF/vPC 等设备级(网络 N:1)虚拟化技术,虽然可以简化拓扑、具备高可靠性,但是对于网络有强制的拓扑形状要求,不够灵活,只适合小规模网络构建,一般只适用于数据中心内部网络。

3) 网络隔离/分离能力限制

当前主流的网络隔离技术为虚拟局域网 VLAN(或虚拟专用网 VPN),在大规模虚拟化环境中部署会有两大限制:一是 VLAN 数量在标准定义中只有 12b,即可用的数量为 4Kb,这样的数量级对于公有云或大型虚拟化云计算应用而言微不足道;二是 VLAN 技术当前为静态配置型技术,使得每个 VLAN 都会通过整个数据中心网络(核心设备更是如此),导致任何一个 VLAN 的未知目的的广播数据会在整网泛滥,无节制地消耗网络交换能力与带宽。

上述的三大挑战来自于物理网络设备本身的限制,因此,为了满足云计算虚拟化的网络能力需求,需要进行大范围的技术革新,在此驱动力基础上,诞生了 Overlay 网络技术。

2. 技术优点

叠加网络是指以现行的 IP 网络为基础,在其上建立叠加的逻辑网络(overlay logical network),屏蔽底层物理网络差异,实现网络资源虚拟化,使多个逻辑上彼此隔离的网络分区以及多个虚拟网络可以在同一物理网络设备上共存。叠加网络已成为当今 SDN 发展的重要动力之一。它的主要思路可被归纳为解耦、独立、控制三个方面[31]。

(1) 解耦:控制与承载分离,将网络的控制从网络物理硬件中脱离出来,交给虚拟化的网络层处理。通过虚拟化屏蔽底层的物理差异,将物理网络资源抽象成网络资源池,正如服务器虚拟化技术把服务器资源转化为计算资源池一样,使得网络资源的调用更加灵活,可以按照用户需求实现资源的按需调配。

(2) 独立:叠加网络方案以 IP 网络为基础,只要 IP 可达,就可以部署相应的虚拟化网络,无须对原有的物理网络架构做出任何改变。

(3) 控制:叠加后的逻辑网络将以软件可编程的方式被统一控制。届时,网络资源、计算资源、存储资源都可以被统一调度,并按照用户需求按需调度。以虚拟交换机为代表的虚拟化网络设备可以利用服务器虚拟化管理程序(hypervisor)实现,也可以以软件方式部署在网关中实现与外部物理网络的整合。各种虚拟化网络设备协同工作,在资源管理平台的统一控制下,通过在节点间按需搭建虚拟网络,实现网络资源的虚拟化。

Overlay 主要具有以下优点：

(1) 基于 IP 网络构建 Fabric。无拓扑限制，IP 可达即可；承载网络和业务网络分离；对现有网络改动较小，保护用户现有投资。

(2) 将用户数从 12b 提高到 16Mb，极大扩展了隔离数量。

(3) 网络简化、安全。虚拟网络支持 L2、L3 等，无须运行 LAN 协议，骨干网络无须大量 VLAN Trunk。

(4) 支持多样化的组网部署方式，支持跨域互访。

(5) 支持虚拟机灵活迁移，安全策略动态跟随。

(6) 转发优化和表项容量增大。消除了 MAC 表项学习泛滥，且东西向流量无须经过网关。

6.7.2 Overlay 技术简介

1. Overlay 概念简介

Overlay 在网络技术领域，指的是一种叠加在物理层网络设备之上的虚拟化技术模式，其大体框架是，基于 IP 网络，对基础网络不进行大规模修改的前提下，应用虚拟网络上的承载，并且每个业务享有独立专用的虚拟网络。其有三大特点：

(1) Overlay 网络是指建立在已有网络上的虚拟网，由逻辑节点和逻辑链路构成。

(2) Overlay 网络具有独立的控制和转发平面，对于连接在 Overlay 边缘设备之外的终端系统来说，物理网络是透明的。

(3) Overlay 网络是物理网络向云和虚拟化的深度延伸，使云资源池化能力不再受到物理网络的限制，是实现云网融合的关键。

Overlay 技术是在现有的物理网络之上构建一个虚拟网络，上层应用只考虑虚拟网络，而不用关心底层物理网络状况。一个 Overlay 网络主要由三部分组成：边缘设备、控制平面和数据平面(如图 6-16 所示)。边缘设备是指与虚拟机直接相连的设备，控制平面主要负责虚拟隧道的建立和维护以及主机可达性信息的通告，数据平面是承载 Overlay 报文的物理网络。

针对前面提到的传统网络三大挑战，Overlay 给出了完美的解决方法。

1) 针对虚拟机规模受限的解决方式

将虚拟机数据封装在 IP 数据包中后，进行数据传递时物理网络只匹配封装后的网络参数，即隧道端点的地址，因此，对于承载网络(特别是接入交换机)，MAC 地址规格需求大大降低，最低规格只需几十个 MAC 地址即可满足需求(每个端口一台物理服务器的隧道端点 MAC)。当然，对于核心设备/网关设备处的表项(MAC/ARP)要求依然极高，当前的解决方案主要采用分散方式，通过使用多个核心设备/网关设备来降低单个设备表项的数量级。

2) 针对虚拟机迁移范围受限的解决方式

Overlay 网络把二层报文封装在 IP 报文之上，因此，只要网络 IP 路由可达就可以部署 Overlay 网络，而目前 IP 路由网络本身已经非常成熟，且在网络结构上没有特殊要求。

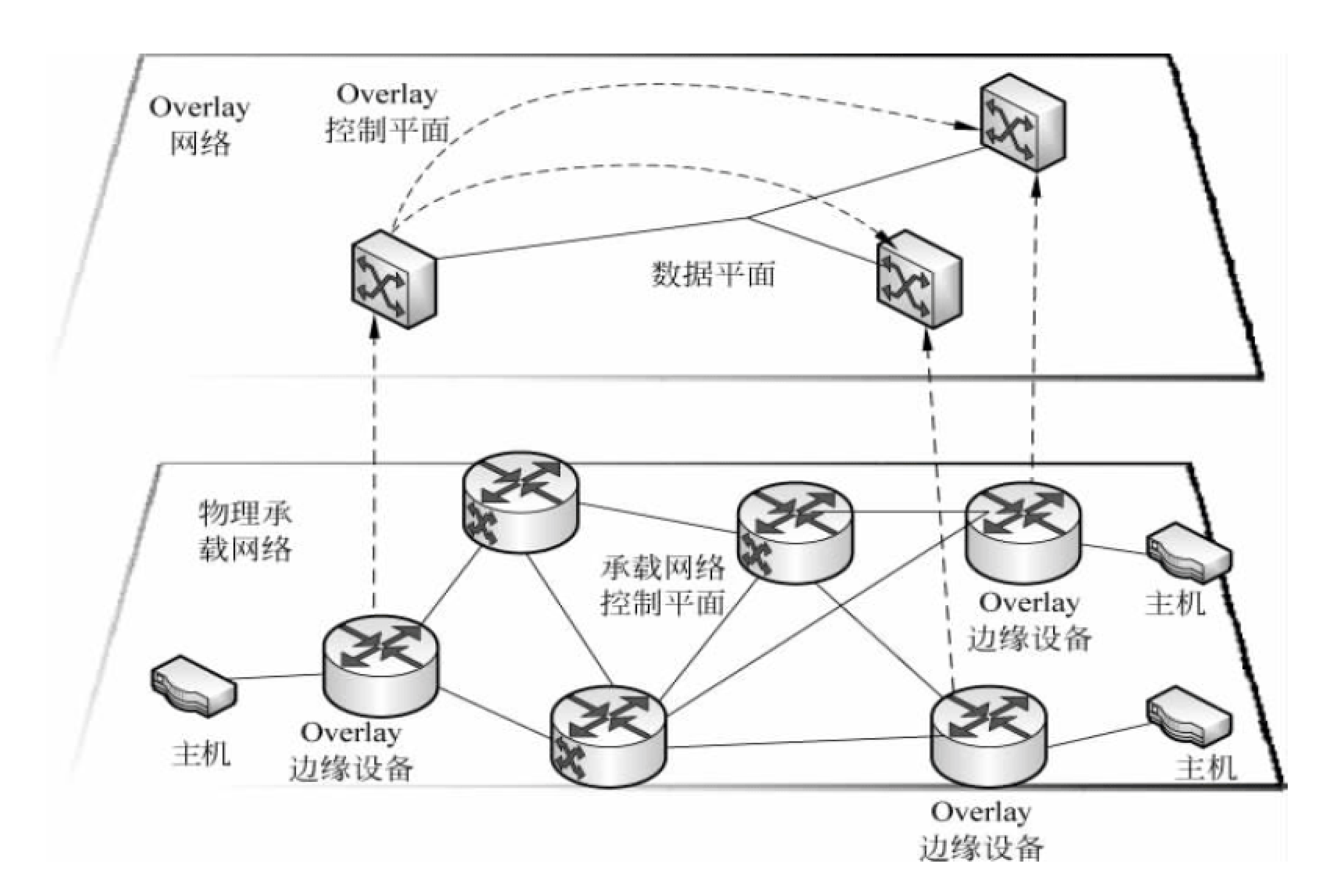


图 6-16　Overlay 架构图

路由网络本身具备良好的扩展能力、很强的故障自愈能力和负载均衡能力。采用 Overlay 技术后，企业不用改变现有网络架构即可用于支撑新的云计算业务，极方便用户部署。

3）针对网络隔离/分离能力限制的解决方式

针对 VLAN 最大只能支持 4K 的容量限制，在 Overlay 技术中扩展了隔离标识的位数，可以支持高达 16M 的用户，极大扩展了隔离数量。

2. 主流 Overlay 技术简介

当前主流的 Overlay 技术主要有 VXLAN、NVGRE 和 STT 这三种二层 Overlay 技术。除此之外，网络虚拟化叠加(NVO3)技术正处于发展阶段，但尚未成熟。

(1) VXLAN 是将以太网报文封装在 UDP 传输层上的一种隧道转发模式，目的 UDP 端口号为 4798；为了使 VXLAN 充分利用承载网络路由的均衡性，VXLAN 将原始以太网数据头(MAC、IP、四层端口号等)的 HASH 值作为 UDP 的号；采用 24 比特标识二层网络分段，称为 VNI(VXLAN network identifier)，类似于 VLAN ID 作用；未知目的、广播、组播等网络流量均被封装为组播转发，物理网络要求支持任意源组播(ASM)。

(2) NVGRE 是将以太网报文封装在 GRE(通用路由封装)内的一种隧道转发模式，主要是微软开发的技术，也是其 Hyper-V 所使用的叠加方法。采用 24 比特标识二层网络分段，称为 VSI(virtual subnet identifier)，类似于 VLAN ID 作用；为了使 NVGRE 利用承载网络路由的均衡性，NVGRE 在 GRE 扩展字段 flow ID，这就要求物理网络能够识别到 GRE 隧道的扩展信息，并以 flow ID 进行流量分担；未知目的、广播、组播等网络流量均被封装为组播转发。

(3) STT(无状态传输隧道)属于 VMware 旗下的 Nicira。STT 是 Nicira 网络虚拟

化平台的一个组成部分，STT 利用了 TCP 的数据封装形式，但改造了 TCP 的传输机制，数据传输不遵循 TCP 状态机，而是全新定义的无状态机制，将 TCP 各字段意义重新定义，无须三次握手建立 TCP 连接，因此称为无状态 TCP；以太网数据封装在无状态 TCP；采用 64 比特 Context ID 标识二层网络分段；为了使 STT 充分利用承载网络路由的均衡性，将原始以太网数据头（MAC、IP、四层端口号等）的 HASH 值作为无状态 TCP 的源端口号；未知目的、广播、组播等网络流量均被封装为组播转发。

（4）NVO3 是由 IETF 的一个工作组开发的。NVO3 与上述叠加技术很相像，也就是说，流量孤立、租户可自由使用所选择的寻址方案，可将虚拟机在网络内自由移动，不必考虑底层核心中三层网络的分隔等。但 NVO3 未来会怎么演进、如何封装尚有待观察。

目前来说，VXLAN 技术相对具有优势，且应用较广。

3. VXLAN 技术简介

Overlay 的本质是 L2 over IP 的隧道技术，在服务器的 vSwitch、物理网络上的技术框架已经就绪，并且从当前的技术选择来看，虽然有多种隧道同时实现，但是以 L2 over UDP 模式实现的 VXLAN 技术具备较大优势，并且在 ESXi 和 Open vSwitch、当前网络的主流芯片已经实现，已经成为主流的 Overlay 技术选择，因此后面的 Overlay 网络均参考 VXLAN 相关的技术组成描述，其他 NVGRE、STT 等均类似。

VXLAN（virtual extensible LAN，可扩展虚拟局域网络）是基于 IP 网络、采用 MAC in UDP 封装形式的二层 VPN 技术。具体封装的报文格式如图 6-17 所示。

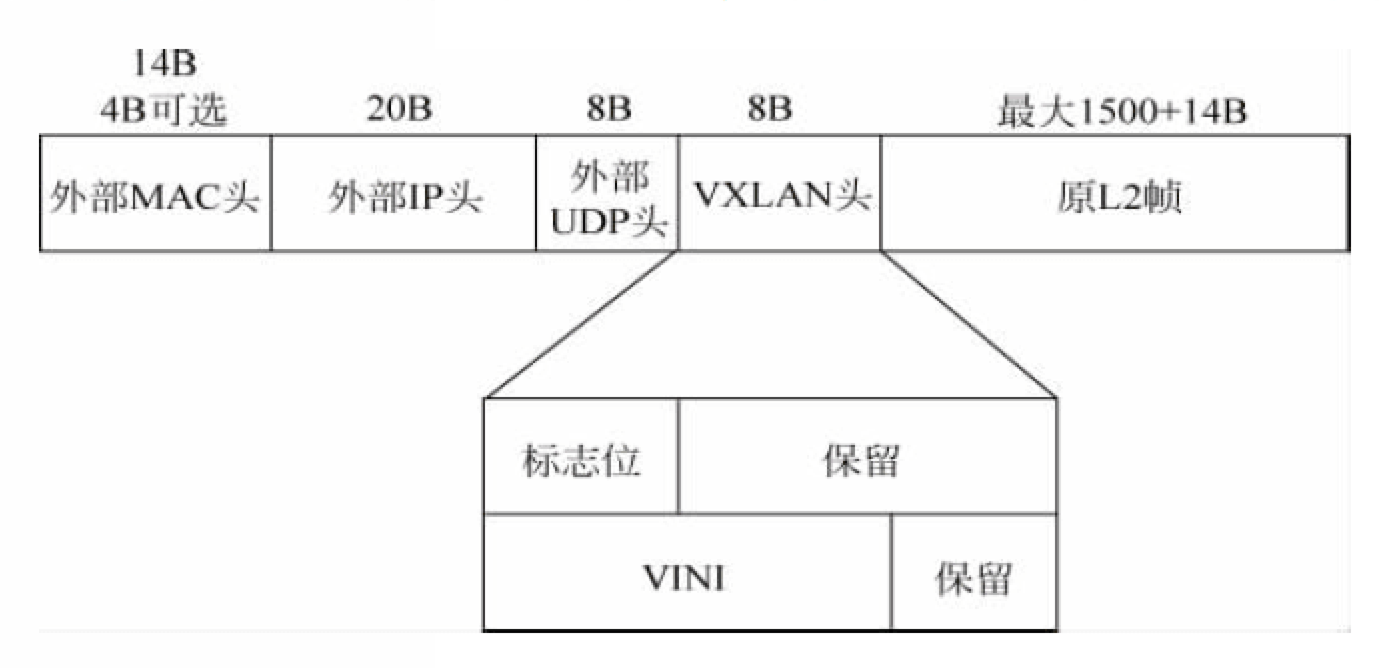


图 6-17　VXLAN 报文格式

VXLAN 网络设备主要有三种角色，分别是 VTEP（VXLAN tunnel end point，VXLAN 隧道端点）、VXLAN GW（VXLAN gateway，VXLAN 网关）、VXLAN IP GW（VXLAN IP Gateway，VXLAN IP 网关），均是物理网络的边缘设备，而由三种边缘设备构成了 VXLAN Overlay 网络，对于应用系统来说，只与这三种设备相关，而与底层物理网络无关。

VTEP 是直接与终端系统连接的设备，负责原始以太报文的 VXLAN 封装和解封装，形态可以是虚拟交换机，也可以是物理交换机。

VXLAN GW 除了具备 VTEP 的功能外，还负责 VLAN 报文与 VXLAN 报文之间

的映射和转发，主要以物理交换机为主。

VXLAN IP GW 具有 VXLAN GW 的所有功能，此外，还负责处理不同 VXLAN 之间的报文通信，同时也是数据中心内部服务向外发布业务的出口，主要以高性能物理交换机为主。

6.7.3 Overlay 组网方案

Overlay 网络架构就纯二层的实现来说，可分为网络 Overlay、主机 Overlay 以及两种方式同时部署的混合 Overlay。Overlay 网络与外部网络数据连通也有多种实现模式，并且对于关键网络部件有不同的技术要求。

(1) 网络 Overlay。在这种模型下，所有 Overlay 设备都是物理设备，服务器无须支持 Overlay，这种模型能够支持虚拟化服务器和物理服务器接入。所以网络 Overlay 的定位主要是网络高性能、与 Hypervisor 平台无关的 Overlay 方案。

(2) 主机 Overlay。所有 Overlay 设备都是虚拟设备，适用服务器全虚拟化的场景，物理网络无须改动。所以主机 Overlay 主要定位是配合 VMware、KVM 等主流 Hypervisor 平台的 Overlay 方案。

(3) 混合 Overlay。物理设备和虚拟设备都可以作为 Overlay 边缘设备，灵活组网，可接入各种形态服务器，可以充分发挥硬件网关的高性能和虚拟网关的业务灵活性。所以混合 Overlay 的主要定位是 Overlay 整体解决方案，它可以为客户提供自主化、多样化的选择。

三种 Overlay 商用模型都通过 SDN 控制器集中控制，实现业务流程的下发和处理，应该说这三种 Overlay 模型都有各自的应用场景。用户可根据自己的需求从上述三种 Overlay 模型和 VLAN VPC 方案中选择最适合自己的模型[31]。

参考文献

[1] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, et al. B4: experience with a globally-deployed software defined wan. [C]. In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM (SIGCOMM '13), New York, NY, USA: ACM, 2013: 3-14.

[2] Hong CY, Kandula S, Mahajan R, Zhang M, Gill V, et al. Achieving high utilization with software-driven WAN. [C]. In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM (SIGCOMM '13), New York, NY, USA: ACM, 2013: 15-26.

[3] 周光涛，郭爱鹏，唐雄燕. SDN/NFV 技术在宽带 IP 城域网中的应用[J]. 信息通信技术，2015(02): 12-15.

[4] 薛淼，符刚. 基于 SDN 的固移网关融合演进研究[J]. 邮电设计技术，2014(05).

[5] Cisco Visual Networking Index. "Global mobile data traffic forecast update. 2013-2018"[J/OL]. http://www.scirp.org/reference/ReferencesPapers.aspx? ReferenceID=1207123.

[6] Prabhu R S., Daneshrad B., " Performance Analysis of Energy-Efficient Power Allocation for MIMO-MRC Systems"[J]. IEEE Transactions on Communications, 2012, 60(8): 2048-2053.

[7] 张健男. 一种基于 Openflow 协议的网络虚拟化平台设计与实现[D]. 北京：北京邮电大学信息与通信工程学院，2014.

[8] 张振海. 异构开放无线网络无缝切换技术研究[D]. 北京：北京邮电大学信息与通信工程学院，2015.

[9] SuKyoung Lee，Kotikalapudi Stiram，Kyungsoo Kim，Yoon Hyuk Kim，NadaGolmie. "Vertical Handoff Decision Algorithms for Providing Optimized Performace in Heterogeneous Wireless Networks"[J]. IEEE Transactions on Vehicular Technology，2009，1(2)：865-881.

[10] Cisco. Cisco Visual Networking Index：Forecast and Methodology，2014-2019[J/OL]. http://www. cisco. com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360. html.

[11] Bianzino AP，Chaudet C，Rossi D，et al. A Survey of Green Networking Research[J]. IEEE Communications Surveys & Tutorials，2012，14(1)：3-20.

[12] D. Pamlin and K. Szomol' anyi. Saving the Climate @ the Speed of Light-First Roadmap for Reduced CO2 Emissions in the EU and Be-yond. [J]. WorldWildlife Fund and European Telecommunications Network Operators' Association，April 2007.

[13] Zhang Y，Chowdhury P，Tornatore M，et al. Energy efficiency in telecom optical networks[J]. Fourth Quarter，2010，12(4)：441-458.

[14] P. Leisching and M. Pickavet. Energy footprint of ICT：Forecasts and network solutions. [C]. In：Workshop at OFC/NFOEC'09，San Diego，USA，Mar 2009.

[15] Vinh Dien HOANG，Xin ZHANG，Stanislav FILIN，Hiroshi HARADA. Introduction to 1900. 7 Network Architecture，Design and Current Status. [C]. In：2012 18th IEEE International Conference on Networks (ICON)，Singapore. 2012.

[16] Alexander Gelberger，Niv Yemini，Ran Giladi. Performance Analysis of Software-Defined Networking (SDN). [C]. In：2013 IEEE 21th International Symposium on Modelling，Analysis & Simulation of Computer and Telecommunication Systems(MASCOTS)，2013，8：389-393.

[17] Yong Cui，Shihan Xiao，Chunpeng Liao，Stojmenovic I，Minming Li. Data Centers as Software Defined Networks：Traffic Redundancy Elimination with Wireless Cards at Routers[J]. IEEE Journal on in Selected Areas in Communications，2013，31(12)：2658-2672.

[18] Dan Li，Yunfei Shang，Wu He，Congjie Chen. EXR：Greening Data Center Network with Software Defined Exclusive Routing [J]，IEEE Transactions on Computers，2015，64 (9)：2534-2544.

[19] Helle B，Seetharaman S，Mahadevan P，Yiakoumis Y，Sharma P，Banerjee S，McKeown N. ElasticTree：Saving energy in data center networks [C]. In：Proc of the USENIX NSDI，California，USA，2010.

[20] Kreutz D，Ramos F M V，Esteves Verissimo P，et al. Software-Defined Networking：A Comprehensive Survey[J]. Proceedings of the IEEE，2014，103(1)：10-13.

[21] Evangelos Haleplidis，Spyros Denazis，Odysseas Koufopavlou. ForCES Applicability to SDN-enhanced NFV. [C]. In：2014 Third European Workshop on Software-Defined Networks (EWSDN)，Budapest，2014.

[22] 王瑞，戴彬，徐冠，等. 面向 SDN 的节能技术探讨[J]. 计算机应用研究，2016，5(33)：1285-1292

[23] 董朝阳，赵俊华，文福拴，等. 从智能电网到能源互联网：基本概念与研究框架[J]. 电力系统自动化，2014，38(15)：1-11.

[24] 人民政协报. 共同推动构建全球能源互联网[J/OL]. http://news. xinhuanet. com/energy/2016-03/03/c_1118220864. htm

[25] 和讯新闻. 国务院发布《关于推进"互联网+"智慧能源发展的指导意见》[J/OL]. http://news. hexun. com/2016-03-10/182688548. html

[26] 周海明，刘广一，刘超群. 能源互联网技术框架研究[J]. 中国电力，2014，11(47)：140-144.

[27] 曹军威,杨明博,张德华,等. 能源互联网:信息与能源的基础设施一体化[J]. 南方电网技术,2014,8(4):1-10.
[28] 王继业,孟坤,曹军威,等. 能源互联网信息技术研究综述[J]. 计算机研究与发展,2015,52(3):1-18.
[29] ZHANG G,SU L,WANG Y. Research on communication network architecture of energy internet based on SDN[C]. IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA),Ottawa,2014.
[30] 王羿. 能源互联网的信息通信架构体系研究[J]. 电力信息与通信技术,2015,13(7):15-21.
[31] SDN Overlay 技术白皮书[J/OL]. http://www.sdnlab.com/16340.html.

第7章 网络虚拟化与网络功能虚拟化

当前，虚拟化的概念大行其道，如货币的虚拟化（比特币）、计算资源和存储资源的虚拟化（云服务）等。虽然，现在的网络技术不断提高，通过光纤等技术，链路的速率等得到了大幅度提升，但是在网络结构或者节点之间的连接关系上，没有比较大的改变。而随着人们对网络需求的提高，当前比较僵化的网络结构已经不能满足人们日益多样的网络需求。网络虚拟化技术慢慢受到了人们的重视。不同于以往的冷冰冰的物理拓扑关系，网络虚拟化所具有的新的特质会更加吸引人，并且在大数据、云计算和物联网等前沿性科技方面，发挥着重要的作用。

另外，传统的移动互联网架构（如3G和4G）空口端，每种网元的功能比较单一，网元之间相互依存、共同配合才能完成数据和信令的传递，却不能相互替代。这是因为，传统的移动网络的网元，硬件实现方式和软件实现方式不同，硬件主要包括主控芯片不同和接口不同。在这种情况下，新一代的网络如5G，不能通过修改上一代的网元来进行网络升级（就如同4G不能通过升级3G网元来升级网络一样）。即使要利用已有的网元，仍然需要通过一些接口转换来实现，这就增加了网络升级的代价。另外，如果通过重新更换软硬件的方法，在造成资源浪费的同时，也会给用户带来不便，因为用户的移动设备都是针对某个移动网络制作的，会产生旧手机不能使用新网络的情况。网络功能虚拟化，能够通过x86的设备设计统一架构和全接口的网元，使网络在升级换代的时候，只需要修改网元中的软件，而无须更换硬件。这样就大大地加快了网络更新频率，并且降低了网络更新过程中的代价。

7.1 网络虚拟化的背景和发展

众所周知，互联网在最初设计时是以互联互通为第一目标，因此才确定了“端到端”的设计原则，即网络提供尽量简单的服务模式，数据的可靠性传输以及其他的应用都由处在网络边缘的用户主机实现。互联网依靠这种简单而实用的技术架构取得了巨大的成功。但是随

着语音、视频等多媒体业务的出现，以及越来越多用户对移动技术的需求，互联网在可扩展性、移动性、服务质量(QoS)、网络安全以及能耗等方面的问题和不足都逐渐体现出来。同时，随着云计算、物联网等新型计算模式与新型业务的不断涌现，现有的 Internet 体系结构已经不适应未来网络的发展需求。互联网的发展[1]逐渐陷入了僵化状态。面对传统互联网体系结构暴露出来的问题，研究学者们开始对现有的网络体系结构进行思考，虚拟化技术一直与网络的发展密不可分，研究学者利用虚拟化技术在已有的网络架构基础上构建出逻辑网络以满足特定的业务需求。而新的技术框架需要新的协议以及新的安全机制，这对网络规划是一个极其重要的挑战。

7.1.1 虚拟化和网络虚拟化技术

虚拟化在信息和通信领域是一个非常宽泛的概念。1959 年，克里斯托弗(Christopher Strachey)发表了一篇学术报告，名为"大型高速计算机中的时间共享"(*time sharing in large fast computers*)，他在文中提出了虚拟化的基本概念，这篇文章也被认为是虚拟化技术的最早论述。最早在商业系统上实现虚拟化的是 IBM 公司在 1965 年发布的 IBM7044。随着 x86 平台处理能力的与日俱增，1999 年，VMware 在 x86 平台上推出了可以流畅运行的商业虚拟化软件。从 2006 年到现在，可以说是进入了虚拟化技术的爆发期。纵观虚拟化技术的发展历史，可以看到它始终如一的目标就是实现对 IT 资源的充分利用。实际上，通常所说的虚拟化是指服务器虚拟化技术。而除此之外，还有网络虚拟化和存储虚拟化技术。其中，网络虚拟化应用于企业核心和边缘路由。而网络虚拟化的发展也有很长的历程。

网络虚拟化技术是指：在公共的网络基础设施上通过抽象、重构、隔离机制来支持多个并行的虚拟网络，各个虚拟网络在自己的资源片上像独享物理网络一样可以根据所提供的业务服务来定制架构与协议；并能够根据网络环境的动态变化对整个网络中的节点资源和链路资源进行灵活调度，从而实现网络资源的合理配置与可控可管，提升网络动态适应性与服务质量保证，实现架构内嵌的安全与信任机制，并降低网络的运营和维护成本。通过网络虚拟化技术来提供网络虚拟化环境，允许多个服务提供商将各自的业务网络实例部署在一个公共的网络基础设施上，动态组合出异构、共存却相互隔离的虚拟网络，可以在这些虚拟网络上部署所需的网络架构，从而支持未来网络"演进"式架构。所以这种网络虚拟化环境一定不能与当前的仅仅提供一种通信隔离的虚拟专有网络技术混淆，网络虚拟化能实现在未来网络架构下将传统的网络与并行的虚拟网络进行便捷而安全的隔离，同时实现各自的管理和控制的自治，以提高未来网络使用网络虚拟化技术的灵活性与可行性。

7.1.2 网络虚拟化的过去

网络虚拟化作为一个单独的概念存在并没有多长时间，但其背后的技术发展却有相当的沉淀。网络虚拟化发展历程如图 7-1 所示[2]。回顾整个互联网的发展过程，就可以清楚地发现人们早就意识到了网络服务与硬件解耦的必要性，也因此诞生了许多过渡的

技术，其中最重要的 4 类分别是虚拟局域网络（VLAN）、虚拟专用网络（VPN）、主动可编程网络（APN）、覆盖网络。

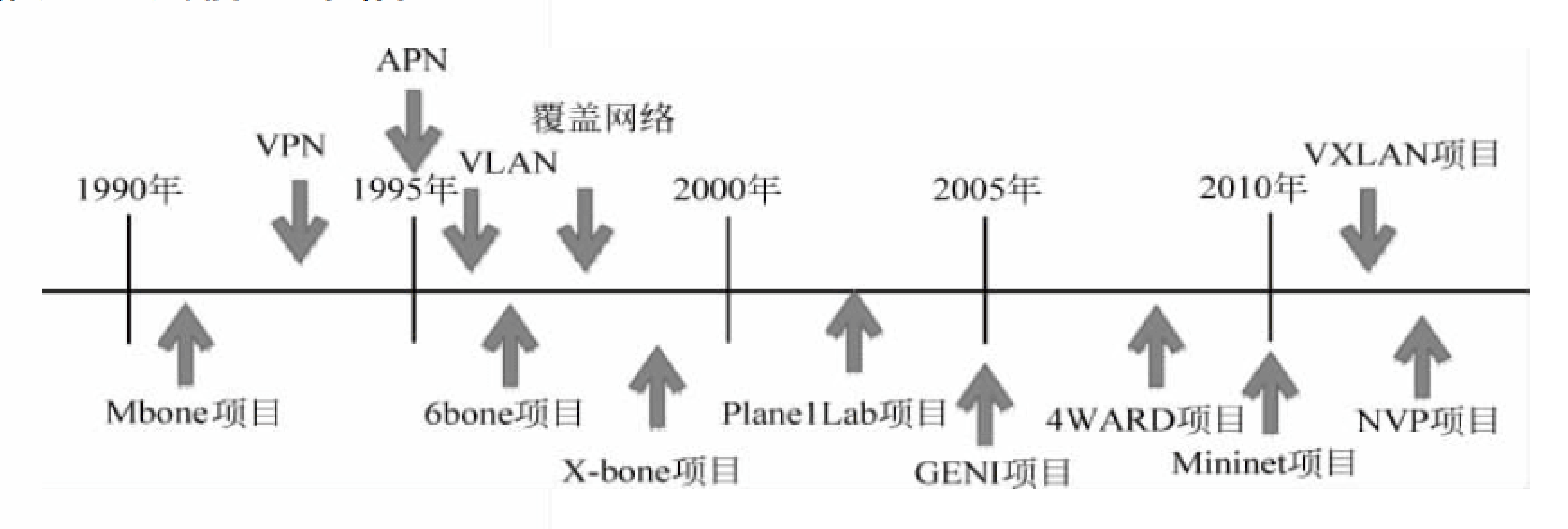


图 7-1　网络虚拟化发展历程

1．虚拟局域网络

VLAN 是一种通过将局域网内的设备逻辑地而不是物理地划分成一个个网段从而实现虚拟工作组的技术。在传统的以太网中，单一的广播域使得网络对于资源的管理手段有限。VLAN 技术的出现使得网络管理人员可以将同一物理局域网内的用户划分到不同的逻辑子网中，具有加强广播控制、简化网络管理、降低建设成本、提高网络安全等方面的作用。VLAN 技术的实现方式也是纷繁多样，包含基于端口、基于媒体访问控制（MAC）、基于 IP 甚至是基于用户自定义的实现方式。IEEE 802.1Q 协议的发布统一了不同厂商的标签格式，进一步完善了 VLAN 的体系结构，也加速了 VLAN 的发展。

2．虚拟专用网络

VPN 是指在公用网络上建立起来的“虚拟”的专用网络，它的任意两个节点间并无传统专用网络所需要的端到端的物理链路，而是构建在公用网络供应商所提供的物理网络之上，通过隧道技术实现站点间的互联，以达到共享物理网络资源的目的，所以说它是一种逻辑网络。VPN 通常用于一些组织或者公司来互连它们的子部门，也可以用于个人远端接入公司内部网络。

3．主动可编程网络

APN 技术是希望将物理网络的资源通过网络可编程接口（NAPI）的形式暴露出来，使得用户可以自定义报文的处理方式。APN 实现方式上主要有两大类，第一类是利用电信技术中的信令方式将网络中传输和控制层面区分开来，抽象出来的控制层面就可以开放网络的可编程接口，允许服务提供者控制网络的状态；第二类则是利用网络本身的资源，将控制信息封装在报文内部，路由器在收到报文时再按照其带内信息处理，达到自定义处理报文的目的。不难看出，第二类在报文级粒度的处理上带给 APN 更多的灵活性，更加适应复杂的网络模型。

4．覆盖网络

覆盖网络是构建在已有网络上的一个逻辑网络，利用隧道或封装等技术将感知节点

互连起来，报文只在感知节点上处理，而在感知节点之外透明地传输。在覆盖网络中，虚拟的是网络的拓扑，所有的感知节点以及它们之间的联系构成了这个逻辑拓扑。覆盖网络技术无须特定的底层网络支持，也不需要改变网络的任何特性，因此常常被用于部署新的网络服务或者优化现有网络服务。

网络虚拟化的概念是由以前研究成果发展而来的，而多个共存逻辑网络在过去关于网络的研究中曾多次出现在文献中。目前的网络虚拟化技术如多协议标签交换(MPLS)和其他类似的技术主要是链路的虚拟化而不是全网络解决方案的虚拟化。而更高级的一些解决方案，如VPN，仅仅是使用公用网络的技术来满足安全、共享、应用的可兼容执行环境，只能提供简单的虚拟连接或IP转发，不能用于端到端的部署或底层基础设施的全虚拟化。重叠虚拟网的许多方案通过引入应用层网络，更接近于全虚拟化的概念。为了支持未来网络“演进”式架构，需要使用网络虚拟化技术提供虚拟网络环境，允许多个服务提供商将许多网络实例部署在一个公共的物理网络设施上，动态组合出异构、共存却相互隔离的虚拟网。但是这种虚拟化网络的类型一定不能与当前的技术混淆，如VPN仅提供一种通信隔离，而虚拟化网络需要在未来网络架构下将IP网络与并行的非IP网络进行自治的管理控制和灵活的安全隔离，以提高使用网络虚拟化作为未来网络的基础的可行性。

7.1.3 网络虚拟化的现在

早期的网络虚拟化的相关工作多在解决一些具体的问题，欠缺完整的技术体系与合理的组织结构。现在的网络虚拟化，一方面借助于云计算的平台来全面展示其优异的性能；另一方面整合自身的体系结构，以平台化的方式提供更为友好全面的服务。除此之外，软件定义网络(SDN)的兴起也为网络虚拟化提供了一个新的思路。下面将重点介绍网络虚拟化与云计算以及与SDN的结合。

1. 网络虚拟化与云计算

一直以来网络虚拟化都缺少一个杀手级的应用，云计算的出现对于网络虚拟化来说是一次千载难逢的机会。首先，云计算对于网络性能以及结构上的复杂要求恰恰是网络虚拟化追求的目标，二者在思想上天然地契合；其次，网络虚拟化与云计算中大量使用的服务器虚拟化可以有效地配合，二者在技术上可以完美地互补。

2. 网络虚拟化平台化

网络虚拟化本身的功能虽然已经相当齐备，然而各类技术都存在着一定的局限性，需要对网络虚拟化技术进行统一整合。这方面的主要成果主要来自产业界，其中Nicira公司的NVP、Cisco公司的OnePK以及Juniper公司的Junosphere平台最具代表性。

3. 网络虚拟化与SDN

网络虚拟化与SDN因为存在许多类似的地方而常常被人混淆，其实SDN要做的是从底层数据平面分离出一个逻辑的中心控制平面，而网络虚拟化是根据逻辑网络对底层

网络进行抽象，两者并非一个概念。不过思想上的相似性确实加强了它们之间的联系。首先，SDN 作为重构网络的技术被认为是实现网络虚拟化的一种有效的手段，如上面提及的 Nicira 公司的 NVP 完全就可以按照 SDN 的方式进行部署；其次，SDN 网络的性能测试可以借助于已有的网络虚拟化工具如 Mininet 实现；最后，网络虚拟化可以通过虚拟 SDN 交换机（事实上就是一些流表）来实现节点虚拟化。

7.1.4 网络虚拟化的目标

网络虚拟化主要的服务对象是人，主要包括普通用户、服务提供商和基础设施提供商。网络虚拟化的目的是按照人的意愿以方便、快捷、高效和安全等原则组建网络。普通用户对网络有安全的需求、专用网的需求（质量保证）、资料分享需求（兴趣爱好）等。

网络虚拟化由服务提供商和基础设施提供商协同完成。服务提供商主要衔接普通用户和基础设施之间的关系，明确普通用户的需求，并将其量化给基础设施提供商。基础设施提供商通过服务提供商提供的需求，组建基础设施网络，在满足服务提供商的需求的同时，降低基础设施网络运行成本。

通过以上分析，我们知道，普通用户是网络需求提出方，是网络虚拟化中最重要需要满足的。服务提供商主要的工作是对用户的需求数据进行分析量化。服务提供商为了更好地、更加智能地满足用户的需求，需要通过大数据等技术对普通用户的数据进行分析和处理，而这些行为，都需要在网络虚拟化过程中预留一些接口，所以服务提供商也有对网络虚拟化的需求。基础设施提供商的主要工作是组建基础网络，同时基础设施提供商自己也要优化基础设施，以高效低成本地运行。所以基础设施提供商也对网络虚拟化有需求。

为了综合满足普通用户、服务提供商和基础设施提供商这三者的需求，网络虚拟化的目标主要有以下几个：

（1）高效率。在网络虚拟化环境中，同一台物理设备上的资源可以服务于多个虚拟网络，可以有效地利用现有资源，发挥虚拟化的高效作用。

（2）好的隔离性。网络虚拟化环境隔离底层物理网络的具体实现，虚拟网络不需要关心底层网络的差异，它们并行运行在统一标准下的虚拟环境中。这样的实现保证了网络的可用性和安全性。与在传统物理网络中运行的应用程序相比，运行于虚拟化环境中的应用程序具有明显更好的性能和效率，采用隔离是主要原因。

（3）高可靠性。网络虚拟化的隔离特点，使得虚拟网络是独立于硬件实现的，是在更高一层的逻辑上运行的。可以通过使用的故障恢复机制来提供虚拟网络业务的连续性和可靠性。当某个物理设备或者物理链路发生故障的时候，可以利用其他物理资源动态恢复在故障资源上的虚拟网络，不影响虚拟网络的运行，有很高的可靠性。

（4）低成本。网络虚拟化具有高效的特点，只需要少量的硬件设备，就能实现在传统网络中需要大量物理网络资源的网络应用，间接地降低了各方面的成本。

（5）好的兼容性。网络虚拟化隔离物理网络的具体实现，同时也提高了网络的兼容性，异构的物理资源可以运行各式各样的虚拟网络。

(6) 易于管理。网络虚拟化可以节省维护服务器所需要的网络管理员数量，对于网络服务器的管理更加简便。

7.1.5 网络虚拟化的未来

回顾发展历程以及当下的研究热点，不难发现网络虚拟化的一个明显的趋势：技术作为支持被模糊化，服务作为核心被抽象出来。这主要是受到云计算中的 XaaS(一切即服务)思想的影响。现在的网络虚拟化还仅仅是为了支持其他服务而存在的底层支撑，它的许多能力因为复杂的特性无法完全展现出来。未来的网络虚拟化应该向着 NaaS(网络即服务)的方向发展，底层的硬件被统一抽象功能化，对外暴露给用户的就是一系列的 API。用户不再像以前只能使用网络，而无法操作网络本身，这对于定制个性化的网络意义重大，而 API 的访问机制也可以保证服务的高效性和安全性。随着工业化与信息化融合进程的快速推进以及物联网、云计算等技术的大规模发展，IP 网络规模、业务规模和用户规模势必将进一步扩大，未来网络虚拟化在打造智能高效的网络服务方面必定大有可为。

网络虚拟化在无线网络和光网络中的推广是其发展的另一个趋势。作为传统网络的重要组成部分，无线网络和光网络的虚拟化必将能进一步提高网络虚拟化的实用价值。不过因为物理资源上的特殊性，它们在网络虚拟化的实现上存在着不小的差异。在无线网络领域，网络虚拟化的实现主要有三种思路：基于数据和流的实现、基于协议的实现和基于频谱的实现。三种思路对虚拟化实现的粒度越来越细，难度也越来越大。在光网络领域，虚拟化的实现则分为三个部分：光交叉连接器(OXC)和单波长级光信号交换的设备(ROADM)的虚拟化、次波长交换虚拟化、光链路的虚拟化。

7.2 虚拟网创建

虚拟网指用户(包括普通用户、运营商和服务提供商、基础设施提供商)对网络的逻辑拓扑结构进行重新组织，以满足自己或者其服务对象的使用习惯、能耗需求、安全需求、速度需求或者用户兴趣爱好分类等。虚拟网在很多资料上也称为网络切片或者切片。

虚拟网的优化指运营商或者网络服务提供商在满足虚拟网服务的基础上，通过一定的技术手段，调整网络元素以及连接关系，以满足网络虚拟化的目标。

网络虚拟化是解决互联网"僵化"问题的有效手段。而虚拟网创建是网络虚拟化的关键目标。网络虚拟化作为构建未来网络的关键技术，旨在构建出完全虚拟化的网络环境，在公共的网络基础设施上通过抽象和重构机制构建出共存但相互隔离的、多样化、异构的虚拟网络。新的网络服务和新的网络协议可以在任意的虚拟网络中运行而不影响其他虚拟网络中的服务。虚拟网络完全控制属于自己的虚拟资源，可以根据自身的业务和需求制定网络架构和网络协议，并可以根据网络环境的动态变化实时调整节点资源和链路资源，从而实现对网络资源的可管可控性。

7.2.1　网络虚拟化结构模型

网络虚拟化结构模型分为有线网络虚拟化结构模型和无线网络虚拟化结构模型。目前的大多数研究都是基于有线网络虚拟化的，随着无线通信技术的发展，用户无线通信设备的增加，无线网络所占的比重越来越大，无线网络虚拟化也越来越得到人们的重视，但是当前的相关研究还很少。

1．有线网络虚拟化结构模型

在当前的 Internet 中，网络服务提供商（Internet service providers，ISP）既负责构建基础设施又负责提供网络服务，这种双重身份阻碍了网络的发展和创新。网络虚拟化环境实现了对 ISP 功能的重新划分和定义，将现有的 ISP 拆分为两个相互独立的实体：基础设施提供商（infrastructure providers，InP）和服务提供商（service providers，SP）。InP 负责建设和管理底层网络（substrate network）资源，InP 通过提供编程接口的方式向 SP 提供服务；SP 从一个或多个 InP 租用底层网络资源，构建和运营虚拟网络（virtual network，VN）并向用户提供网络服务，不同的 SP 通过共用现有的底层网络资源来创建不同类型的虚拟网络。虚拟网络由虚拟节点（virtual node，VN）和连接这些节点的虚拟链路（virtual link，VL）组成，每个虚拟节点一般对应于底层的一个物理节点，虚拟链路对应于底层节点间的一条路径。SP 可以完全控制自身所构建的虚拟网络，即可以对虚拟网络使用不同的网络类型的架构、不同的路由协议、不同的转发方式等。终端用户（end user，EU）可以有选择地接入一个或多个虚拟网络中。

图 7-2 表示网络虚拟化环境中网络运营模式与传统互联网结构中网络运营模式的区别。由图可知，在传统的网络运营模式中，用户一般只能接入一种 ISP 中，可选的服务较少；在网络虚拟化环境中，用户或虚拟运营商接入 SP 运营的虚拟网络，而 InP 只是负责

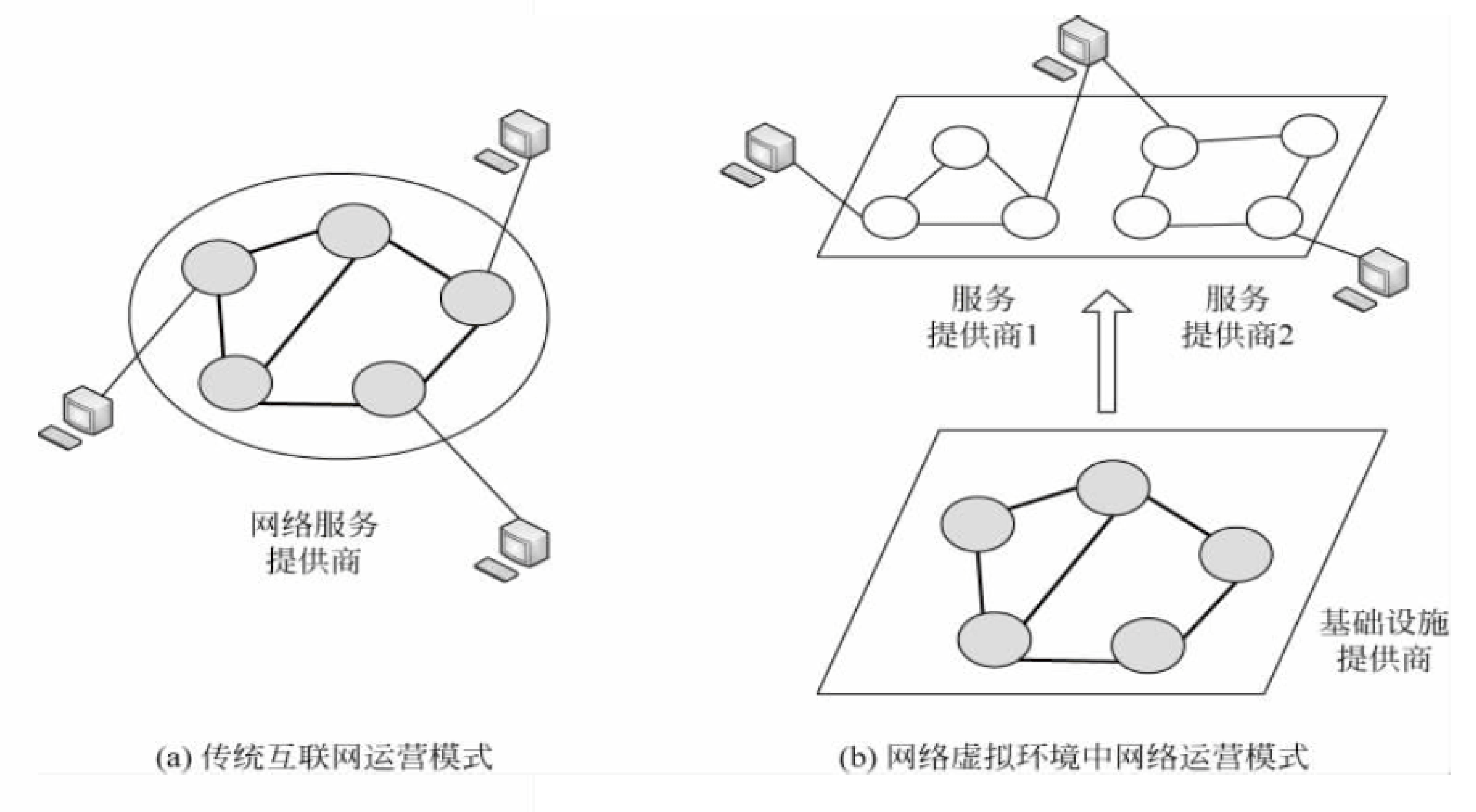


图 7-2　网络运营模式的变化

维护和建设底层网络并向 SP 提供物理资源，多样化的虚拟网络为用户的选择提供了更多可能。

基础设施提供商、服务提供商和用户是网络虚拟化环境中的三种主要角色，下面分别对这三种角色的功能进行详细描述。

1）基础设施提供商

在网络虚拟化环境中，基础设施提供商构建和管理底层网络，并开放编程接口，向不同的服务提供商提供物理资源。不同的基础设施提供商在资源数量、位置和覆盖范围等方面存在区别。基础设施提供商作为物理资源的提供者需要具有以下功能：

（1）虚拟底层资源（节点、链路等）并保证虚拟资源间的排他性和隔离性。

（2）开放功能接口，定义服务提供商的访问权限和访问范围，保证物理资源的安全性。

（3）管理功能。实时管理和监视物理资源的运行情况，将资源使用情况向服务提供商通告。

（4）计费功能。按照虚拟资源的使用数量和使用时间向服务提供商收费。

2）服务提供商

服务提供商从一个或多个基础设施提供商租用物理资源建立虚拟网络，部署和定制网络架构和相应的网络协议，为用户提供端到端的服务。服务提供商可以自定义虚拟网络的数据类型、转发机制，并可以根据用户的需求和数量动态地增加和减少虚拟网络资源。在网络虚拟化环境中，服务提供商作为实际的网络服务提供者需要具有以下功能：

（1）动态地申请、修改和撤销虚拟资源请求。

（2）通过基础设施提供商开放的接口访问虚拟资源，创建虚拟网络并向用户或者虚拟运营商提供服务。

（3）监测和管理虚拟网络。

3）用户

在网络虚拟化环境中，由于存在多类别的虚拟网络，用户的选择更加灵活和自由。用户可以同时接入多个虚拟网络中（传统互联网中的多家乡），根据业务的不同自动选择不同的服务提供商。对于 FTP 等带宽需求较大的业务，用户可以选择提供高带宽的虚拟网络提供商，而对于视频、音频等对时延敏感的业务，用户可以接入提供低时延的虚拟网络提供商。运营商也可以根据用户需求选择所需的虚拟网络，企业或者校园网络可以有自己的专有虚拟网络。

网络虚拟化后带来以下好处：

（1）构建多元化的、异构的下一代互联网成为可能。下一代互联网的关键是支持异构网络并存，SP 专注于运营和维护网络，完全可以根据业务需求制定相应的网络协议，而不需要担心底层网络结构的影响。同时，虚拟网络共存但互相独立的机制使得快速部署新业务成为可能，降低了对已有网络服务的影响，加快了部署周期。

（2）虚拟网络的跨 InP 部署降低了地域性对网络服务的影响。传统互联网受限于多运营商的管理，各个运营商之间的网络策略和管理很难保持一致，严重影响着用户网络服务质量。网络虚拟化环境中的 SP 可以从多个 InP 申请和租用资源，在虚拟网络中，SP 完全可以采用相同的网络策略，从而提升端用户在网络中的体验。

（3）降低部署和运营成本。由 InP 提供物理底层资源，可以有效减少对资源的重复

部署，另外，通过虚拟化技术向多个 SP 提供服务，提高了物理资源的利用率也降低了 SP 的运营成本。

2. 无线网络虚拟化结构模型

网络虚拟化技术是有线领域研究的一个热点，而无线的网络虚拟化技术[3]是一个全新的领域。无线网络虚拟化通过资源虚拟化和控制虚拟化，将传统静态封闭的网络转变为开放智能融合网络。虽然现在无线网络虚拟化还没有一个统一认可的标准，但是诸如软件定义网络(SDN)、软件定义无线电(SDR)以及认知无线电等技术的发展已经为无线网络虚拟化所面临的挑战提供了解决思路。无线网络虚拟化技术的提出为异构无线网络提供了一种有效管理方式，通过对网络资源的抽象和统一表征、资源共享和高效复用，实现异构无线网络的共存与融合。无线网络虚拟化可使复杂多样的网络管控功能从硬件中解耦出来，抽取到上层做统一协调和管理，从而降低网络管理成本，提升网络管控效率。下面是无线网络虚拟化结构模型。

1）接入网虚拟化

国际标准化组织 3GPP 的系统架构工作组已经开展符合技术演进的虚拟化架构——无线接入网共享增强(RSE)，并定义了多个运营商共享无线接入网(RAN)资源的场景，提出了网关核心网(GWCN)和多运营商核心网络(MOCN)两个参考框架。接入网虚拟化框架如图 7-3 所示[3]。GWCN 方案通过多个运营商共享移动性管理实体(MME)来实现移动性管理和承载管理等功能的共享。MOCN 方案中各运营商采用各自完整的核心网，仅在 eNodeB 层面进行资源共享。

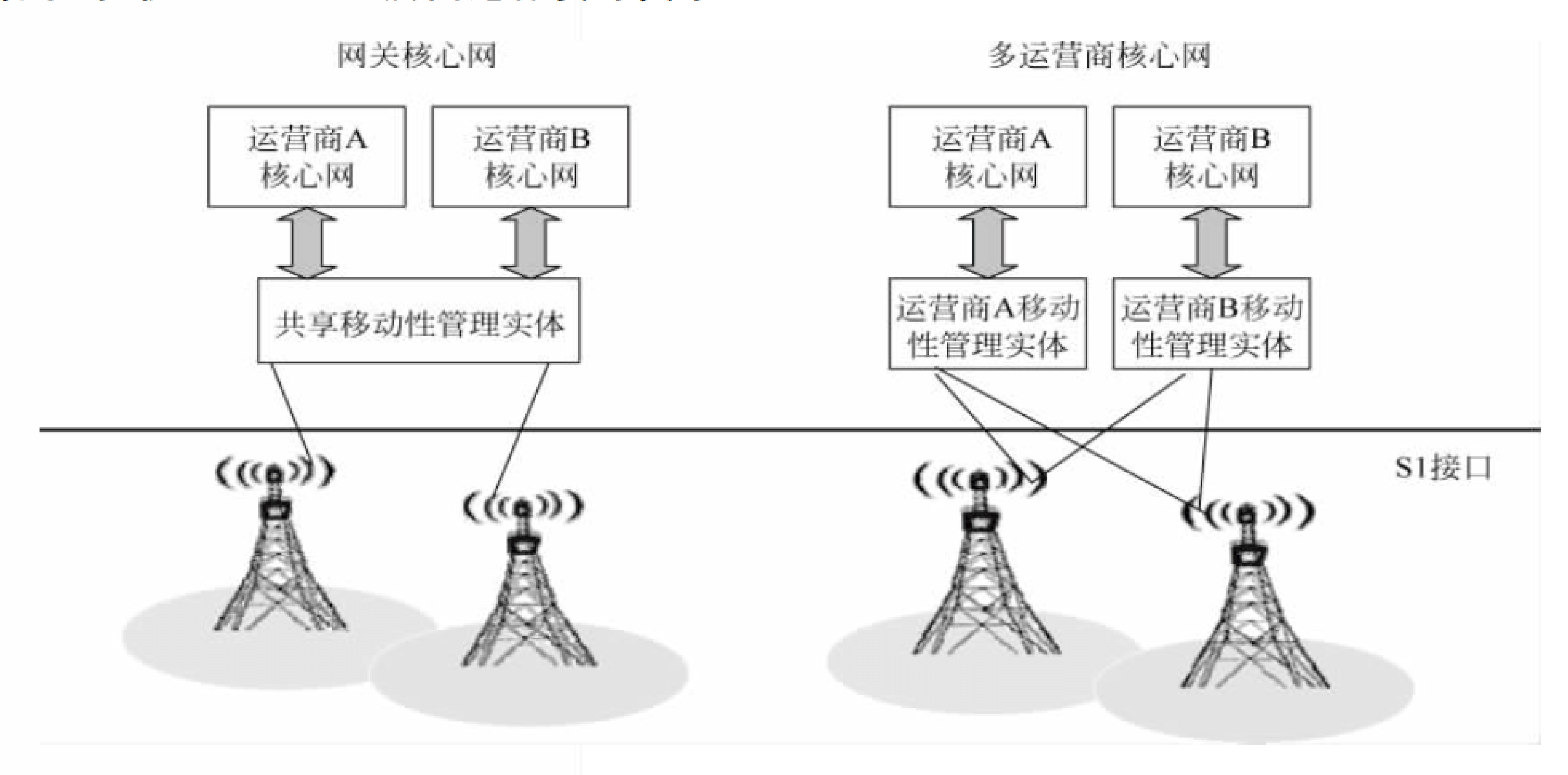


图 7-3　接入网虚拟化架构

3GPP 组织定义接入网虚拟化的目标在于通过共享接入网络资源，提升多个无线网络资源整体利用效率，满足数据业务爆炸式增长所带来的容量增长的需求。主要包含以下几方面：

(1) 根据相关的共享协议和/或政策，能够使无线网络资源在网络实体间进行共享。

(2) 根据不同无线接入共享场景，能够高效地共享无线接入资源。

(3) 针对更细时间粒度下的需求，能够灵活和动态地分配无线接入资源。

(4) 根据相关的共享协议和/或政策，能够合理高效地解决网络过载问题。

针对以上需求，RSE 的实现需要以下 4 种功能：

(1) 无线接入网拥有者一方面需要允许共享接入网资源的参与运营商（如虚拟运营商）获得相应的无线接入网络资源，另一方面也需要这些参与运营商有同等机会获取无线接入网拥有者对网络操作管理与维护的状态信息。

(2) 参与运营商可以根据对网络容量需求的变化，提出不同的接入网络资源需求，以满足业务需求，例如虚拟运营商在工作忙时需要更多网络资源满足容量需求，而在晚上或者周末等闲时仅需要占用少量无线接入网络资源。

(3) 无线接入网拥有者根据参与运营商的容量需求变化，自主重配置无线接入网络资源，尽量满足共享网络的业务 QoS。

(4) 无线接入网拥有者需要依据参与运营商的资源分配情况和网络负载情况，执行合理的负载均衡措施。尤其当小区出现过载情况，无线接入网拥有者可以根据每个参与运营商可承受的最大负载情况，合理地将用户卸载到其他小区。

以上是 3GPP 组织提出的无线接入网共享增强 RSE 方案，通过自主动态地调整无线接入网络资源，以实现接入网资源的有效共享。这种共享方式可以有效地带动产业界和运营商的商业模式转变。此工作已由 TR22.852 完成，并将开始新一轮后续研究工作。

2) 全网虚拟化

无线全网络虚拟化中网络可以由服务提供商（SP）和基础设施提供商（InP）组成。基础设施提供商负责生产和管理从接入网到核心网的整个网络的基础设施，譬如基站设备等，服务提供商负责为用户提供多样化的业务。基础设施提供商的资源往往虚拟化为多个子部分，服务供应商根据用户需求，请求相应的子部分资源，为终端用户提供端到端服务，并忽略底层物理网络结构的差异。这样每一子部分都认为其本身是一个完整的网络系统，包含虚拟核心网和虚拟接入网，这些子部分也称为虚拟网络。如图 7-4 所示[4]是服

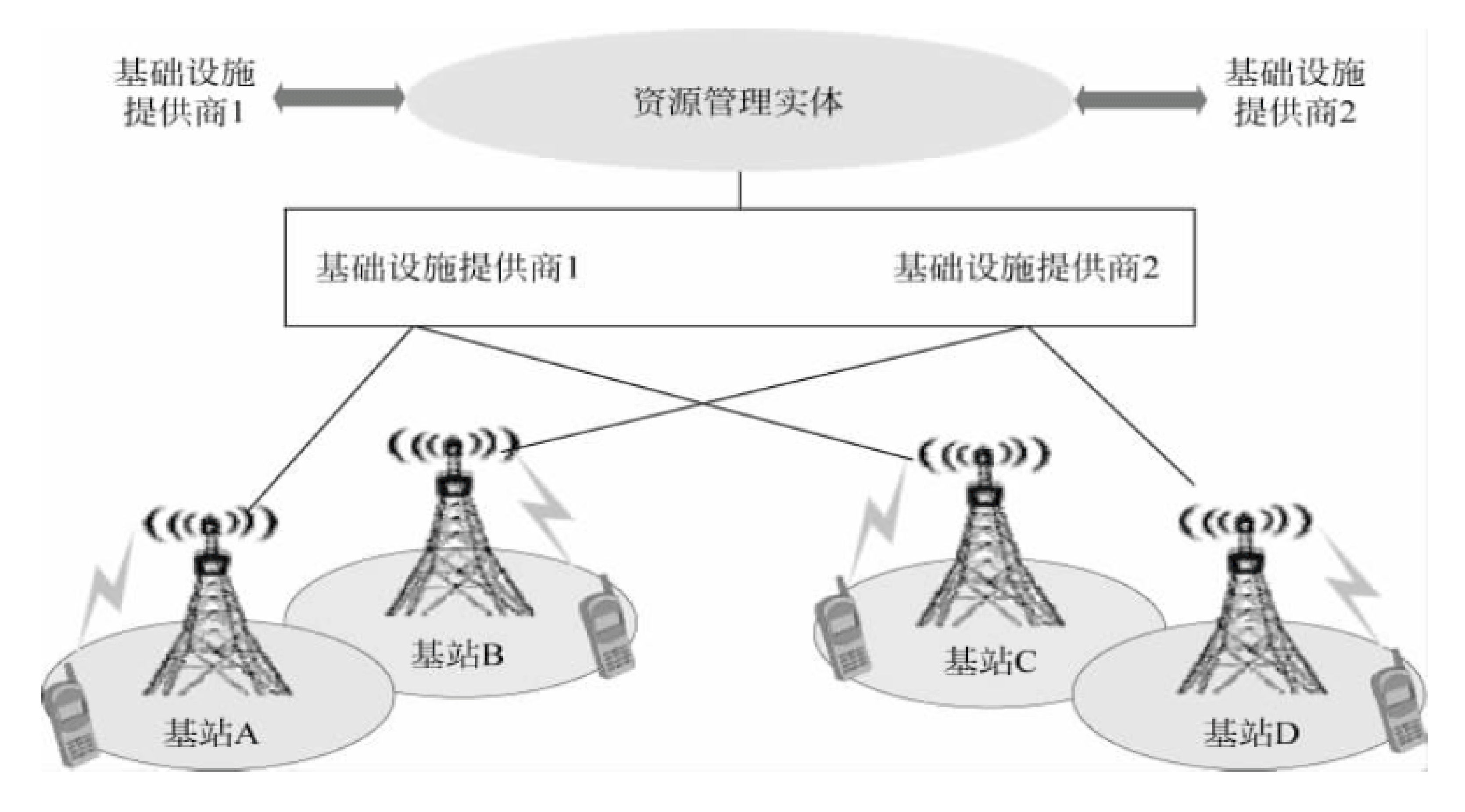


图 7-4　SP 和 InP 实现无线网络虚拟化的一种网络架构

务提供商和基础设施提供商实现无线网络虚拟化的一种网络架构。如果设备商在同一个区域内有共同的基础设施(基站)覆盖,则其资源可以被不同的服务商共享使用。图中,基站 A 和基站 B 属于 InP1,基站 C 和基站 D 属于 InP2,上层的资源管理实体采用集中方式来管理 InP 之间的协作和隔离。重叠覆盖的基站(如 A 和 B)为服务提供商按需提供物理资源,这种方式淡化了网络基础设备和设备商的隶属关系,而变为资源池的方式供 SP 按照最优资源分配或最小代价来使用。但是这种网络结构需要明确不同 InP 的基站覆盖范围,否则,会由于服务提供商用户不被 InP 覆盖造成服务空洞。

SP 和 InP 也可分别称为移动虚拟运营商(MVNO)和移动运营商(MNO)。很多国家已经颁布政策要求运营商开放其无线接入网络资源给虚拟运营商,鼓励多元化市场竞争。这样一方面移动运营商可以通过将无线接入网络资源租赁给移动虚拟运营商获取利润,同时也为自身腾出空间发展最擅长的通信技术,研发基站设备和实现更加精细化的运营维护等。另一方面,移动虚拟运营商可以发挥其市场优势,投入更多精力在用户市场需求分析预测、新型增值业务和功能化业务的开发与推广等工作,为用户提供更为专业和定制化的服务,在获取市场利润的同时,提高无线网络空闲资源的利用效率,进一步提升整个无线网络容量。

7.2.2 虚拟网创建流程

虚拟网创建[5]是整个网络虚拟化的重点问题,其决定者是用户,或者是网络提供者依据自己的学习能力自动优化网络。用户按需请求,虚拟网络提供商根据请求来调配底层资源。虚拟网创建流程如图 7-5 所示[5]。

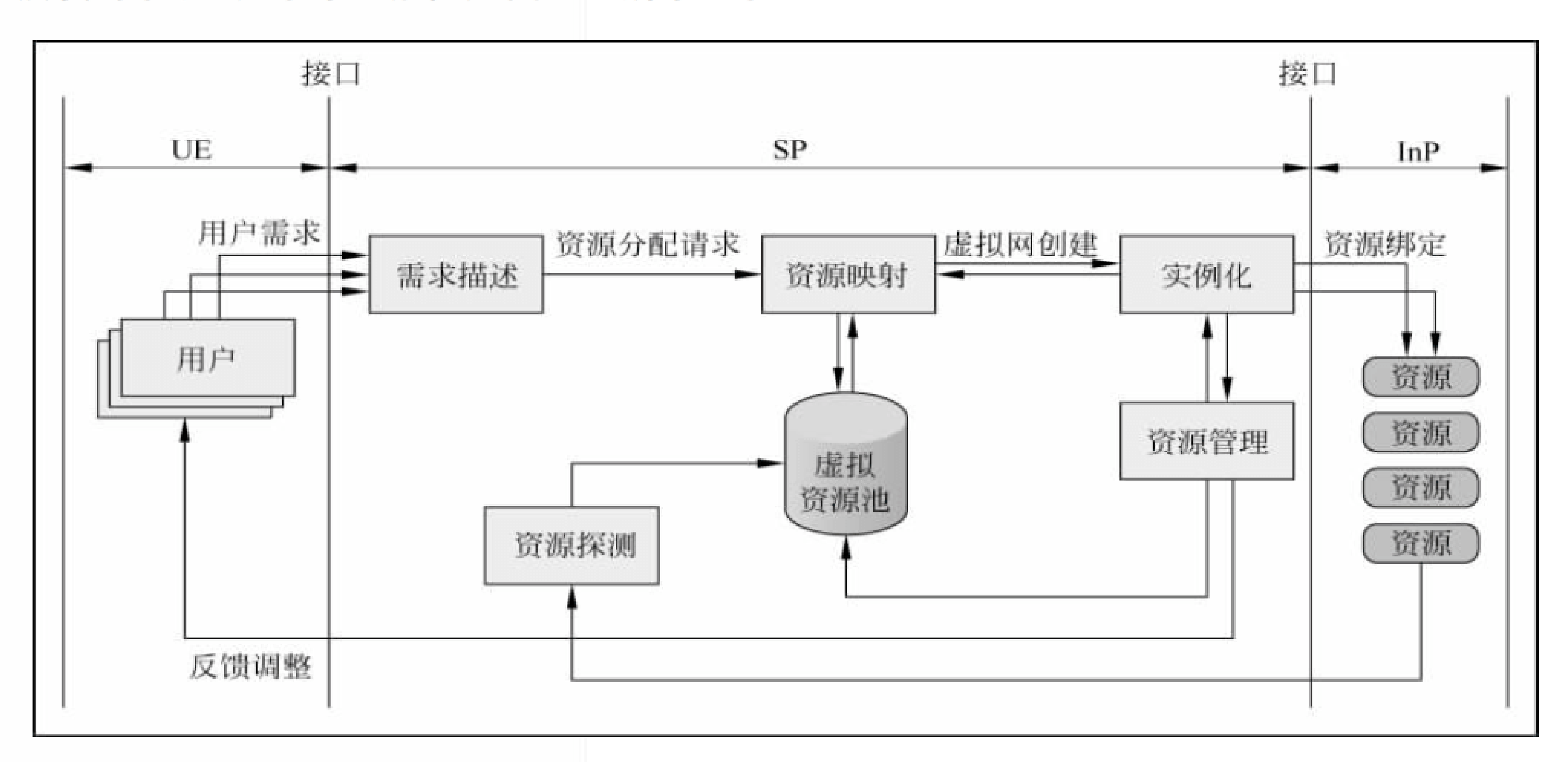


图 7-5　虚拟网创建流程

首先终端用户发起资源请求,通过统一资源描述的方式对用户需求与物理资源的描述进行相似度匹配,其中需求的描述包括节点的数目、处理能力、链路的带宽、拓扑信息等。目前的虚拟资源描述解决方案多是用可扩展标记语言(XML)进行描述,从而满足灵

活性、可扩展性、层次性、安全性等需求。

其次进行虚拟资源的映射，也称为“虚拟网的嵌入”。也就是在 UE 将需求准确地描述给 SP 后，SP 将这些虚拟网络请求进行划分，分别映射到不同的 InP 上，这也是虚拟网创建过程中最重要的环节。资源映射属于多目标多约束问题。一方面，要同时满足用户的节点、链路的需求；另一方面，要使得资源的开销最低、InP 收益尽量最大化，还要兼顾负载均衡等问题。目前基于代理的集中式和基于策略的分布式映射算法是域间虚拟网络映射的热点研究问题。这个过程涉及虚拟资源的动态探测，通过某一 SP 周期性地扫描本地管理的资源，将资源的属性和状态收集、储存起来，放到虚拟资源池中，并以列表的形式招贴出来，供其余的 SP 查看、调度。虚拟网络映射算法要解决将虚拟网络资源映射到物理资源之上，也就是网络资源应用虚拟化的问题。它的主要目标是将虚拟网络请求合理分配到实体网络中并同时满足节点和链路的约束。

最后是虚拟网的实例化构建。SP 选定可用的虚拟资源后，进行资源绑定而实现虚拟网络的实例化，将用户需要的虚拟环境安装到底层的设备中，在虚拟机中部署用户需要的协议、软件等，如支持某种路由协议的 XORP 软件，然后将这些资源连接起来，形成用户需要的特定虚拟网络。在虚拟网进行实例化后，用户可以通过 SP 提供的接口（Web 接口或 SSH 接口等）对虚拟网络进行管理和控制。同时规划单元负责虚拟网状态的监控，按照用户反馈的信息对虚拟网进行动态的局部调整，如增加或者删减某些节点，扩大或者缩小链路的带宽等。在虚拟网的生存周期结束时，SP 将会注销虚拟网，收回绑定的资源，并更新虚拟资源池中的资源状态。下面分别介绍 SP 中的模块。

1. 需求描述

需求描述是对现有数字资源的结构，以用户使用习惯、用户需求与用户分类等多种分析模型进行综合评估；依据不同评价分析标准、符合良好用户体验标准与分类的多维度需求进行逻辑上的切分进行的描述。

2. 资源映射

1）有线网络资源映射

（1）资源映射流程。整个资源（虚拟网络）映射流程可以归纳为 5 个步骤：请求到达、请求分类、资源更新、响应请求、完成分配。映射流程如图 7-6 所示[7]。

图 7-6 中的资源分配过程进行了时间窗的划分，时间窗决定了分配过程运行的频率，它独立于 InP 的操作决策。基于时间窗的基础，收集到达的虚拟网络请求，然后进行分类、排序，最后放进请求队列中进行相应的处理。由于在一定的时间窗内不可能满足所有到来的虚拟网络请求，因此需要制定相应的请求服务策略机制加以控制。

（2）资源映射模型。如图 7-7 所示[7]，存在两个基础设施提供商的两个独立的物理网络，在这两个物理网络之上实现了两个并行的虚拟网络，同时要对 5 个终端用户提供网络服务。在实际的商业环境中，多个基础设施提供商的竞争是在所难免的，在这个例子中用两个不同的物理网络来表征这样的竞争关系。底层物理网络中的每个物理节点只能支持同一个虚拟网络中的一个节点，但是可以支持不同虚拟网络中的多个节点。虚拟网 1（VN1）和虚拟网 2（VN2）分别由两个网络服务提供商 SP1 和 SP2 建立。网络 1 和

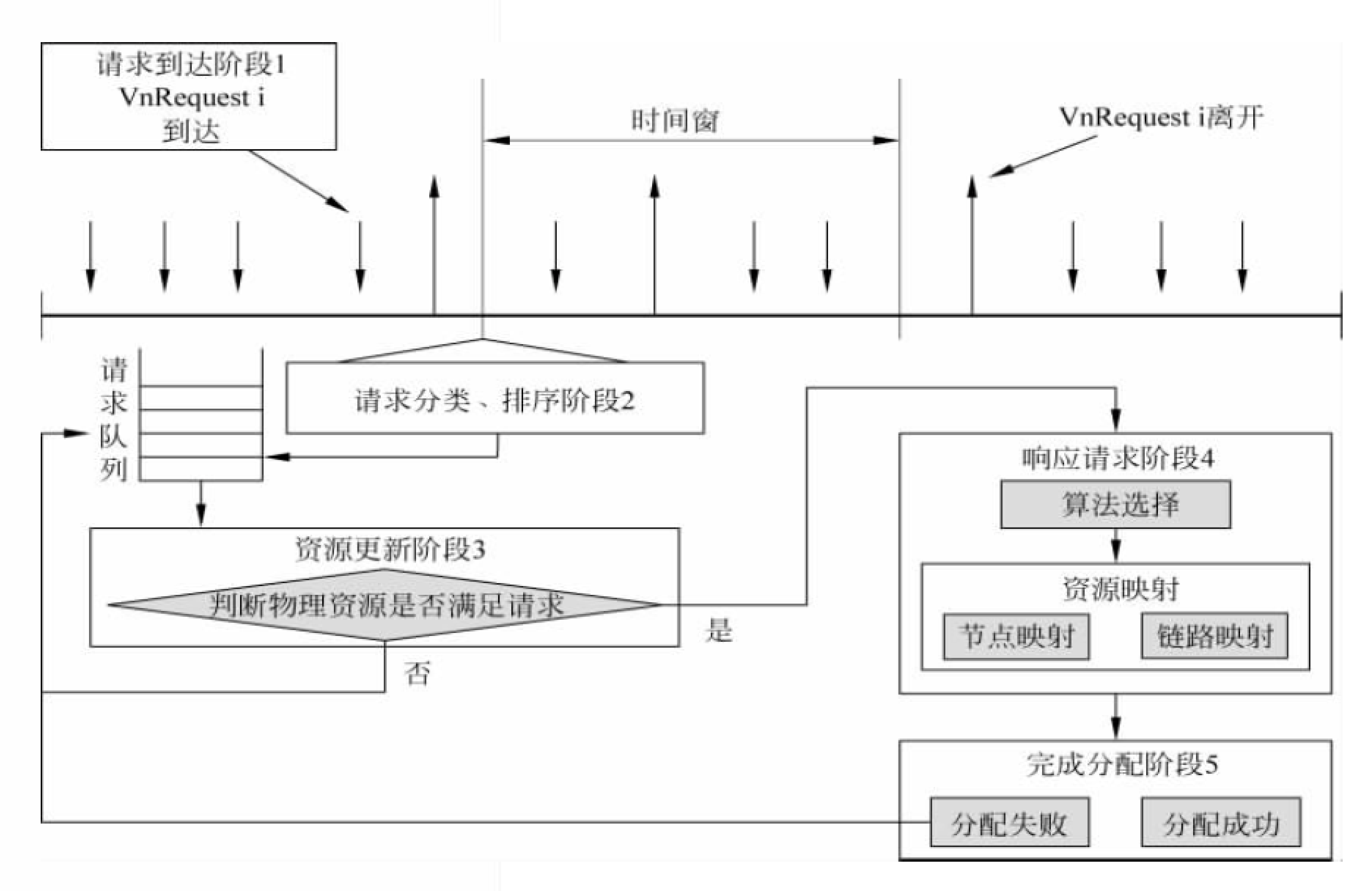


图 7-6　虚拟网资源映射流程

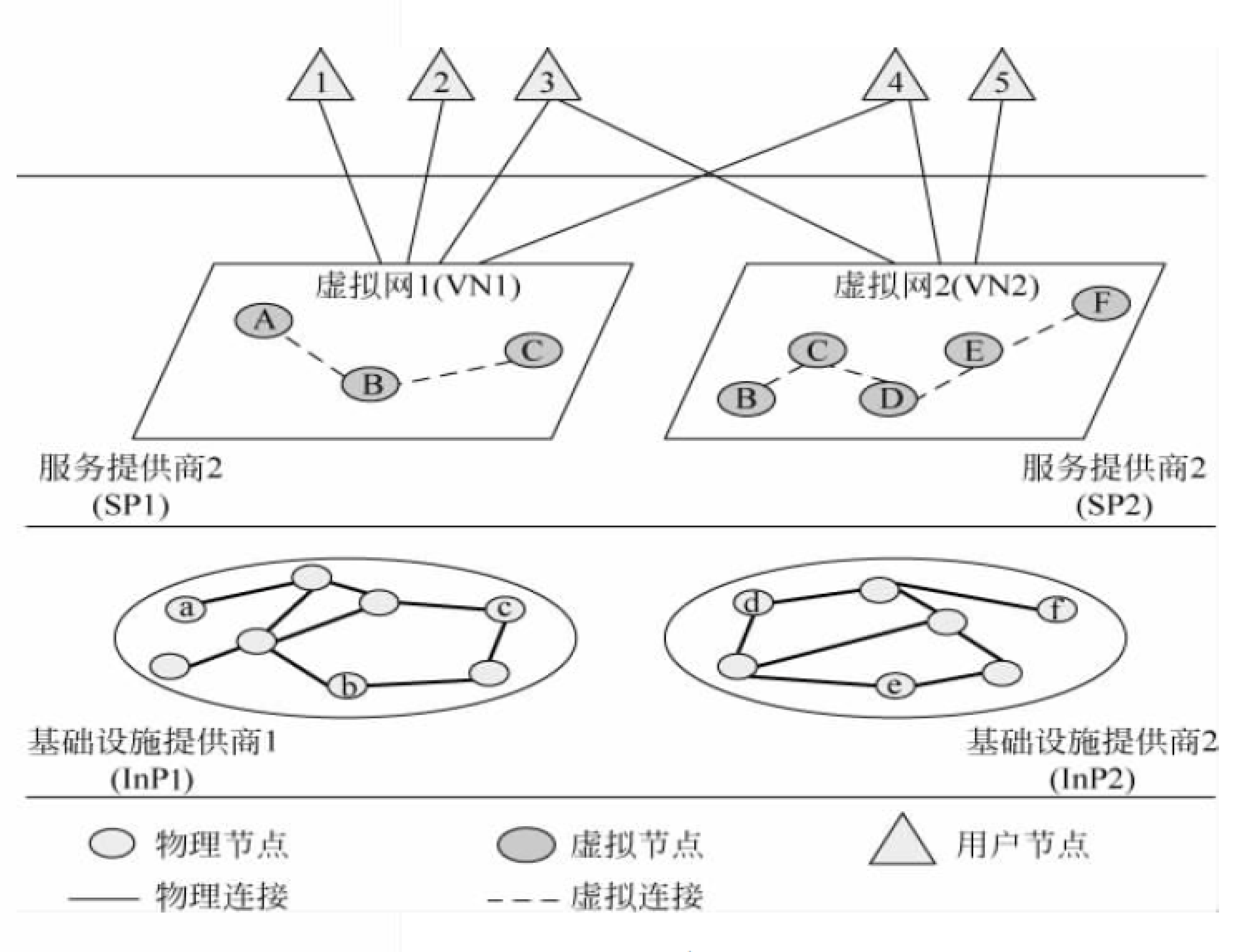


图 7-7　网络虚拟化映射简单模型

网络2的虚拟节点能够同时映射到物理节点b和c上。物理链路也能同时支持多个虚拟网络链路。终端用户可以自主选择不同的虚拟网络,每个虚拟网络共享多个物理网络的资源。服务提供商SP1租用基础设施提供商InP1的物理资源创建虚拟网1,并面向终端用户U1、U2、U3、U4提供网络服务;服务提供商SP2租用基础设施提供商InP1和InP2的物理资源创建虚拟网2,并面向终端用户U3、U4、U5提供网络服务。

一个虚拟网络是由虚拟节点和虚拟链路构成的集合。虚拟网络映射,或称虚拟网络嵌入,它的主要目标是将虚拟网络请求合理分配到实体网络中并同时满足节点约束和链路约束。其中,节点约束的定义是,虚拟节点所映射的实体节点的CPU计算能力必须满足不小于这个虚拟节点的CPU计算能力需求;而链路约束的定义是虚拟链路所映射的实体链路的带宽容量必须满足不小于这个虚拟链路的带宽容量需求。如果整个底层物理网络是有多个基础设施提供商提供的,建立虚拟网络需要横跨多个异构物理网络,则需要在各个物理网络之间提供协作。

例如有一个这样的虚拟网络,每个节点需要800MHz的CPU计算能力。每条链路都需要不低于2Mb/s的带宽流量;地理位置约束,每个虚拟节点和物理节点都有自己的位置属性,有些情况下虚拟节点会优先映射到距离较近的实体节点上;访问控制,在虚拟网络请求很多而物理网络资源无法同时满足需求的情况下,必须对虚拟网络请求有选择性的舍弃策略。虚拟网同样会有多种拓扑类型结构,虚拟网络映射算法可以利用不同的拓扑结构进行有策略的映射。

(3)资源映射问题数学描述。在虚拟网络映射问题[7]中,底层物理网络拓扑被抽象定义成一个无向图G^S,其定义为$G^S=(N^S,E^S,A_N^S,A_L^S)$。其中$N^S$和$E^S$分别代表底层物理网络(实体网络)的节点集合和链路集合。A_N^S表示物理网络的实体节点的资源属性,即实体节点的可用CPU容量;而A_L^S表示底层物理网络的实体链路限制条件,即实体链路的可用带宽容量。此外,每个实体节点$n^S\in N^S$都有一个位置属性$\log(n^S)$,而所有的实体路径集合表示为P^S。

相似的,所有的虚拟网络请求也将被抽象为一个无向图G^V,其定义为$G^V=(N^V,E^V,C_N^V,C_L^V)$。其中$N^V$表示虚拟网络请求所有节点组合的集合,$E^V$表示虚拟网络请求所有链路的集合。$C_N^V$表示虚拟网络请求的虚拟节点的节点限制条件,即虚拟节点的CPU容量;而C_L^V表示虚拟网络请求的虚拟链路限制条件,即虚拟链路的带宽容量。每个虚拟节点$n^V\in N^V$都有一个位置属性$\log(n^V)$。

基于上面的两种拓扑网络的定义,虚拟网络嵌入可以定义为从一个虚拟网络请求G^V映射到实体网络G^S上的一个子集上的映射操作M。该子集是G^V在G^S中的像,$M_N:N^V\to N^S$表示节点映射,$M_E:E^V\to P^S$表示链路映射。其中,任何一个虚拟节点的映射结果是一个实体节点;在同一个虚拟网络的映射中,任意两个不同的虚拟节点不同同时映射到同一个实体节点上;任意一个虚拟链路的映射结果是一条实体路径,而且这条路径的起点和终端也是这个虚拟链路的起点和终点的映射节点。映射操作M可用公式表示如下:

$$M(G^V):(G^V,C_N^V,C_L^V)\to(G^S,\mathrm{A}_N^S,\mathrm{A}_L^S)$$

该定义可以进一步分解为两个部分,节点映射M_N和链路映射M_E:

$$M_N(G^V):(N^V,C_N^V)\to(N^S,A_N^S)$$

$$M_E(CG^V):(E^V,C_L^V)\rightarrow(E^S,A_L^S)$$

一条虚拟网链路可以映射到由多跳实体链路组成的一条路径上，每条实体路径均须满足虚拟网络链路带宽要求 C_L^V。

(4) 资源映射算法的性能评价标准。虚拟网络映射的目标是映射的收益最大化和成本最小化，从而提高映射算法的收益成本比。提高虚拟网络映射的接受率也是需要关注的目标。这个性能指标的定义如下：

收益：收益是用来判断嵌入算法性能的主要因素之一，因为它代表了虚拟网络映射算法的成效。

支出：成本是嵌入的代价。它是根据实体网络中节点的计算能力和链路的带宽被虚拟网络请求占用的情况来定义的。

收入支出比(R/C)：收益成本比 R/C 表征实体网络资源的使用效率。它是判断嵌入算法性能的一个重要指标。收益成本比 R/C 是一个 0 到 1 之间的浮点值，1 表示最优化的嵌入。

虚拟网络请求接受率(acceptance ratio)：虚拟网络请求接受率表征所有虚拟请求成功映射的比率，代表了虚拟网络映射算法性能的一个评价标准。

运行时间(runtime)：运行时间是指一定数量的虚拟网络请求进入虚拟嵌入算法到完成映射关系的寻找，并建立了正确的网络虚拟映射到实体网络中。一个网络窗口的平均运行时间，就是总的运行时间除以网络窗口的数量。运行时间越短，说明算法越快、越高效。

(5) 资源映射算法的分类。虚拟网络映射算法[8]是一种资源调度算法，要解决将虚拟网络资源映射到物理资源之上，也就是网络资源应用虚拟化的问题。它的主要目标是将虚拟网络请求合理分配到实体网络中并同时满足节点和链路的约束。按照不同的底层网络架构，网络映射算法可以划分为两种类型：有线虚拟化映射和无线虚拟化映射；针对不同的资源映射方式，虚拟资源映射算法包括静态资源映射算法、动态资源映射算法、自动资源映射算法三类。静态资源映射算法是指每个虚拟网络请求时被分配固定的资源且在虚拟网络整个生命运行阶段不再改变，如果虚拟网络请求的资源超出了剩余的底层物理资源就会被设为等待甚至拒绝。动态资源映射算法是指在虚拟网络运行的阶段会根据指定的固定的时间间隔对资源重新进行调整分配，如在虚拟请求过载的情况下会进行重新分配以提高网络的性能。两者的区别在于，后者可以根据当前的业务需求和服务性能自适应地调节资源分配，而前者是不允许进行资源分配的任何改变。自动资源映射算法指的是设计一个监督模块和自动分配模块对底层的物理资源进行自动分配。

有线虚拟化资源映射算法如下：

对于有线虚拟化资源映射算法的研究，研究者主要集中在静态资源映射算法和动态资源映射算法上，而自动资源映射算法只涉及概念的层面。动态资源映射算法较之于静态资源映射算法具有更强的复杂性，但自适应强，在平均网络负载和网络性能上，具备更好的优化性。

下面分别介绍这些算法。

静态资源映射算法：静态资源映射算法体现了“一次分配，终身受用”的分配特征。

虽然静态资源映射算法设计复杂度较低且效率高，但不能根据虚拟网络的资源需求以及底层物理网络的动态性进行自适应反应。因此基于无重配置的静态资源映射算法通常采用优化的启发式算法进行求解。静态资源映射被看作是离线的负载均衡问题，可以转换为 NP 难的不可分流问题。静态资源映射算法可分为“二分步”“协分步”和“一步”映射算法。

“二分步”映射算法特点如下：

“二分步”映射算法是指虚拟网络映射过程中将节点和链路映射相分离，即首先对虚拟节点的优化映射，然后基于定位好的虚拟节点，寻找优化的虚拟链路映射，在寻找优化解的过程中，通常需要根据优化目标采用贪婪的策略进行优化求解。“二分步”映射算法虽然简化了资源分配的复杂性，但节点和链路的分配彼此相分离。因此，不具有协调性或者协调很差、效率很低、速度不快，并且难以得到虚拟网络分配的全局最优。例如，非重配置的虚拟网络映射算法 VNA-I，其基本思想是首先将虚拟节点映射到低负载的物理节点上，并且保证分配的物理节点是距离已经分配的节点较近，所有的虚拟节点映射完之后，链路映射阶段采用的是最短路径算法进行物理路径选择。

“协分步”映射算法特点如下：

“协分步”映射算法是指节点和链路分配具有很强的相互协调性分配，即节点分配阶段充分考虑链路分配阶段，以达到全局最优的映射效果，是一个 NP 难问题。例如，一种基于 QoS 保障的虚拟网络分配算法 QoSMap，算法利用一条中继路由节点为虚拟链路提供高质量的后备路由路径，提高虚拟网络的服务质量和可靠性。该算法的映射流程为：首先从未映射的虚拟节点中选取度最大的节点，为其在底层网络中寻找满足其度需求的所有底层节点，其次根据底层节点所能提供的后备路由路径的数量，底层节点是否已经在虚拟网络的后备路由路径中，以及与底层节点相连的链路服务质量三个因素来确定该虚拟网络节点的映射方案。如果在映射过程中 QoSMap 算法不能从底层网络中寻找到能够满足虚拟节点度需求的底层节点，则回溯到上一个虚拟节点映射阶段，然后再对该节点进行重映射。

“一步”映射算法特点如下：

“一步”映射算法是指节点和链路的分配是在同一阶段完成，在映射节点的同时完成了链路的分配。“一步”映射算法是一个 NP 难的问题，通常是采用启发式算法与其他一些理论相结合进行求解。“一步”映射算法不仅能实现收益率更高的虚拟网络映射，而且能达到更快的映射速度和效率。

动态资源映射算法：动态性包含底层物理网络的动态性、虚拟网络请求的动态性以及虚拟网络运行过程的动态性等多方面。针对“一次分配，终身受用”的静态映射算法存在灵活性弱、自适应性差、网络负载不均衡、资源利用率低等问题，研究者提出了动态资源映射算法。动态资源映射算法能够自适应虚拟网络请求和底层物理资源的实时变化，动态地调整节点和链路的负载达到网络负载的均衡性，保证网络的畅通运作，提高对底层物理资源的利用率以及优化网络的服务性能。然而，动态资源映射算法在进行资源分配调整时，势必会增加网络重配置的开销，甚至造成网络服务的中断，同样会影响到网络的实际性能。

2）无线网络资源映射

目前很多接入和核心网络都采用无线通信技术，因此无线虚拟化资源映射算法的研究具有重要的意义。然而，作为一种全新的领域，无线虚拟资源映射算法的研究较少。在无线底层物理网络中构建虚拟网络实例的主要目标是设计可行的、可靠的、高效的无线虚拟化资源映射算法，充分地利用底层无线网络中稀缺的物理资源（节点能力、链路容量、频谱和功耗等）。

无线网络与有线网络最大的区别在于无线传输链路易受环境影响而对信号造成衰减。由于无线链路具有广播性质，一个节点发出的无线电信号可以被其他多个节点获取。因此，需要通过时间、频率或者码字等不同的维度来区分无线信号，以降低多条无线链路之间的干扰。在无线网络虚拟化中，虚拟化节点和链路也需要通过不同的维度（时间、频率、空间、码字等）来避免不同虚拟链路之间的干扰。无线网络资源虚拟化的关键问题是如何将网络底层各个维度资源与网络需求相匹配。网络底层资源可以分为多个正交的维度，例如，时间、频率和空间等，并可以定义每个维度的能力大小。简单地说，如果有一个虚拟化无线网络同时支持频分复用（FDD）和时分复用（TDM）方式，包括频域和时域两个维度。当一个无线节点采用 802.11b 协议传输时，其频域上的能力值为 3，因为 802.11b 协议支持 3 个互不干扰的正交信道，而时域上的能力值需要根据单位时隙的长度和子帧长度来确定。虚拟网络对资源的需求也可以划分为多个维度，而且网络底层的各维度必须要大于虚拟网络对各个维度资源的需求。

3. 资源管理

资源管理系统[6]就是为众多研究人员或团体开展形式多样的创新网络试验提供授权认证的、安全隔离的、可控可管可测的、自适应的试验平台资源共享机制。其基本功能是接受虚拟网络构建请求，根据当前可用资源信息以及虚拟网络对资源的需求信息，以满足研究人员请求的最佳资源组合创建“切片”，即虚拟网络交付给研究人员使用。在试验平台中资源管理主要基于集中式模式，也有一些提出了针对虚拟网络构建的分布式资源管理模式以及两者结合的资源管理模式。下面分别予以讨论。

1）有线网络资源管理

（1）集中式资源管理。集中式资源管理框架一般包括管理服务、管理核心、基底资源三部分。管理核心负责对用户、组件、集合和切片等的注册登记，在用户提出构建“切片”请求时，进行资源发现操作，告诉用户哪些资源可用，然后根据用户需求和用户的证书为用户创建相应的“切片”。一般情况下，集中式资源管理架构实现起来简单高效，但集中式资源管理模式的可扩展性、单点故障与自适应能力方面存在诸多问题。扩展性方面，由于基础设施逐步虚拟化，虚拟资源规模将迅速扩大、类型将更加丰富，这就容易造成资源管控与调配上的性能瓶颈，同时构建虚拟网络的并发性增加会导致虚拟网络请求的接入率下降；同时集中式管理中心容易单点故障而造成系统瘫痪，故健壮性不强。

（2）分布式资源管理。分布式资源管理系统不再有一个管理控制中心指导映射的过程，各物理节点以自组织、自管理和相互协同的方式实现虚拟网络映射和相应的资源分配，因此分布式系统可以有效解决虚拟网络映射方面资源管理分配的可扩展性问题。但

由于虚拟资源的动态性以及自主物理节点各自调整负载导致资源分配的复杂化，同时由于资源状态数据的分散性，不能保证在多个自主物理节点之间进行调度时数据的一致性，导致这种分布式管理模式的健壮性及稳定性不好。同时该架构需要依赖物理节点间大量的消息交互才能获得当前物理网络的可用资源状态及当前虚拟网络的映射状态，难以从宏观层面掌握所有的自主物理节点的状态，因此，映射的全局优化性能没有保证。自主节点间的通信协议复杂，交互信息量大，对网络造成额外的负载，影响网络的性能，尤其在大型网络中应用效果差。

（3）集中式分层树状管理架构。在虚拟化网络分层架构中，虚拟化管理层充当管理平面的角色。集中式管理平面主要由一个统一的中心来进行管理、决策和控制，具有全局优化的资源利用效率，但是控制处理都由管理中心实现，容易造成负载过载，同时效率相对比较低，适合规模较小的网络环境，不利于扩展；分布式管理平面由于没有中心控制而具有良好的扩展性能，但是分布式通信协议、同步机制的实现都非常复杂，额外的通信开销给网络带来负载，且在跨域资源管理调度中同步没有保证。鉴于集中式与分布式机制的优劣，如图 7-8 所示[6]，提出了一种基于虚拟化管理层的网络虚拟化资源管理的集中式分层树状架构。

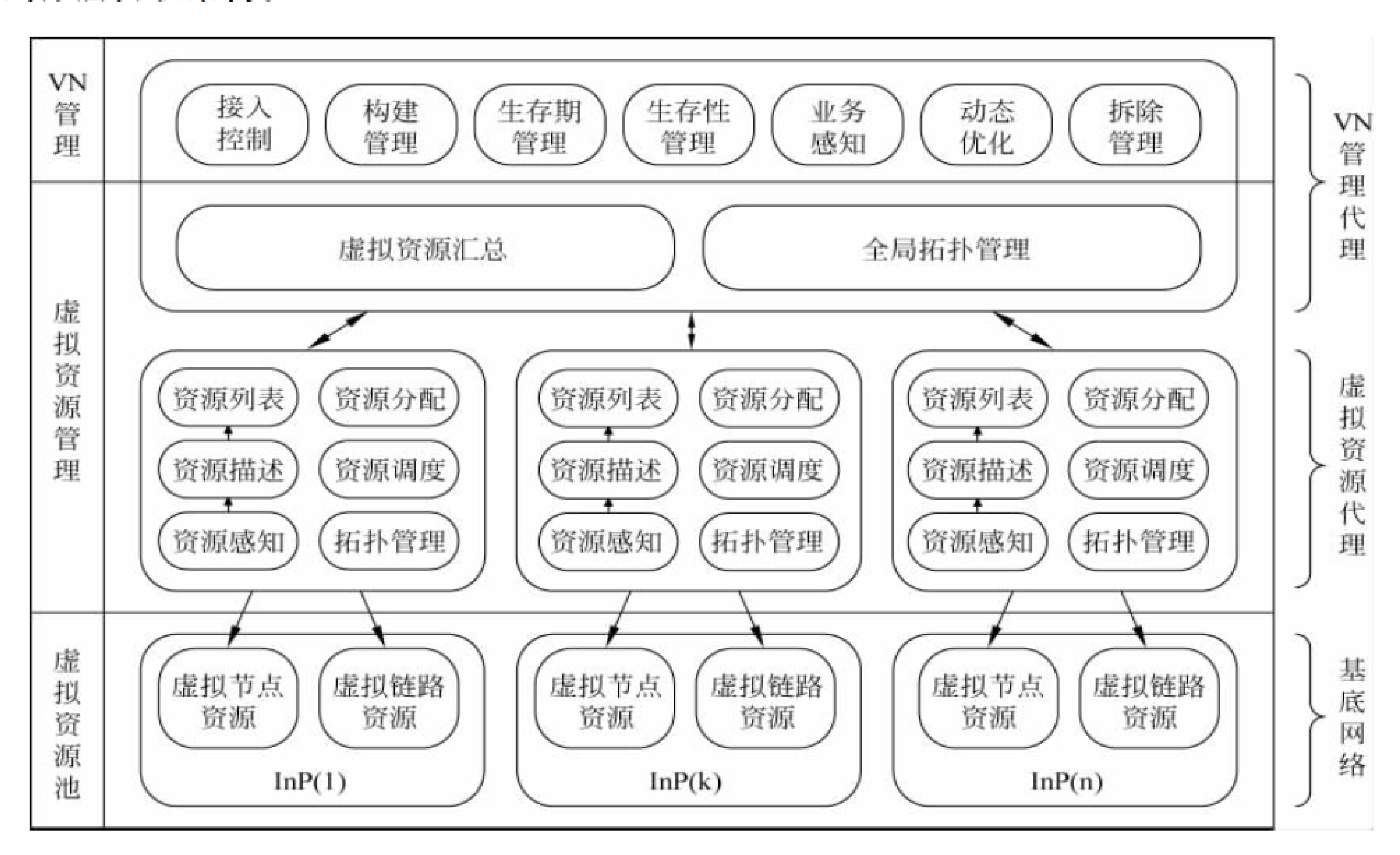


图 7-8　虚拟化管理层架构

首先介绍一下在虚拟化管理层的两个功能实体：虚拟资源代理和 VN 管理代理。每个 InP 构成基础设施层中的一个域，每个 InP 域对应一个虚拟资源代理。虚拟资源代理主要负责区域内虚拟化资源的管理和控制，一方面虚拟资源代理执行资源感知，发现资源并对资源进行统一描述形成资源列表，把资源列表注册到 VN 管理代理，资源感知还包括资源状态监测、故障管理及移动性跟踪；另一方面，虚拟资源代理要根据 VN 管理代理规划后的决策对虚拟资源执行相应操作，主要包括资源分配、资源调度。VN 管理代理

则汇总各虚拟资源代理的资源列表信息并形成全局网络资源信息，在全局信息基础上对允许接入的虚拟网络请求的实例化或是运行中虚拟网络间动态资源分配进行优化规划形成资源配置决策，并将决策通过指令通知各相关虚拟资源代理对相应域资源进行控制操作来实现优化的分配与调度；同时 VN 管理代理还要负责虚拟网络的管理，包括虚拟网络请求的接入控制、虚拟网络拆除、业务感知等功能。

下面通过虚拟网络实例化来说明两个功能实体的工作流程：

① 上层 SP 向 VN 管理代理发起构建虚拟网络的申请，进入虚拟网络构建请求队列中排队。

② VN 管理代理首先根据接入控制策略对队列中的请求进行调度选择，然后根据选中的虚拟网络请求的实例化需求与汇总的网络虚拟资源供给状况进行统一规划得出优化的映射方案。

③ VN 管理根据这个方案向涉及的虚拟资源代理下达资源控制指令，由相关的虚拟资源代理来分别对映射方案进行部署，当实施部署完毕，虚拟资源代理与 VN 管理代理同步对数据库中相应的资源列表项进行更新。

④ 由虚拟化管理层进行调度、创建、分配，选择适当的"切片"资源实例化出满足基本要求的虚拟网络来定制网络并开展业务和服务。

当然，VN 管理还要对运行中的虚拟资源代理和 VN 管理代理进行业务感知，实时做出调整资源调度分配的策略。从整体上讲，虚拟化管理层的两个功能实体，VN 管理代理是集中式的，处在管理架构的顶层，掌握着所有虚拟网络的管理并在汇总了各 InP 域虚拟资源代理的资源列表信息后对资源在虚拟网络间的分配与调度进行优化的统筹规划，是决策者的角色；而各域虚拟资源代理相对底层 InP 域内分布式的虚拟化资源是集中的，但相对上层 VN 管理则是分布式的，负责收集域内虚拟资源的信息并汇报到决策层和根据上层决策对域内虚拟资源执行相应控制操作，是执行者的角色。这一系列分布式虚拟资源代理与集中式 VN 管理代理就构成了集中式分层树状资源管理架构，它的基本特征是"资源的集中管理，分布控制；决策的统一规划，分头部署"。

2）无线网络资源管理

认知无线电技术可提升通信系统的频谱资源利用效率。国际上许多学者都认为动态频谱管理技术是支撑无线网络虚拟化资源管控的有效手段之一。在认知场景下，非授权用户通过协商或者机会式的方式接入授权频段，但是必须保证不影响授权用户的正常通信。由于无线环境的复杂性，用户接入频段会受到来自时域、空域等多种因素的限制，为了保障用户的服务质量，需要为用户分配合适频段来避免干扰。下面提出集中式和分布式两种动态频谱管理方法，解决无线网络虚拟化中频谱资源高效分配与管理难题。集中式网络动态频谱管理如图 7-9 所示。分布式网络动态频谱管理如图 7-10 所示[3]。在多种接入技术（RAT）共存和重叠覆盖的无线环境下，两种方式通过新添加的功能模块完成网络上层的频谱资源分配操作。

基于所提出的动态频谱管理架构，无线网络基站可以选择灵活的频谱资源共享，保证有足够的频谱资源满足用户业务需求，而不必考虑上层的频谱资源是如何分配的。无线网络基站通过对两个网络的业务量进行分析预测为两个网络动态地分配频谱资源。实验验证可提升频谱利用效率 30％以上。

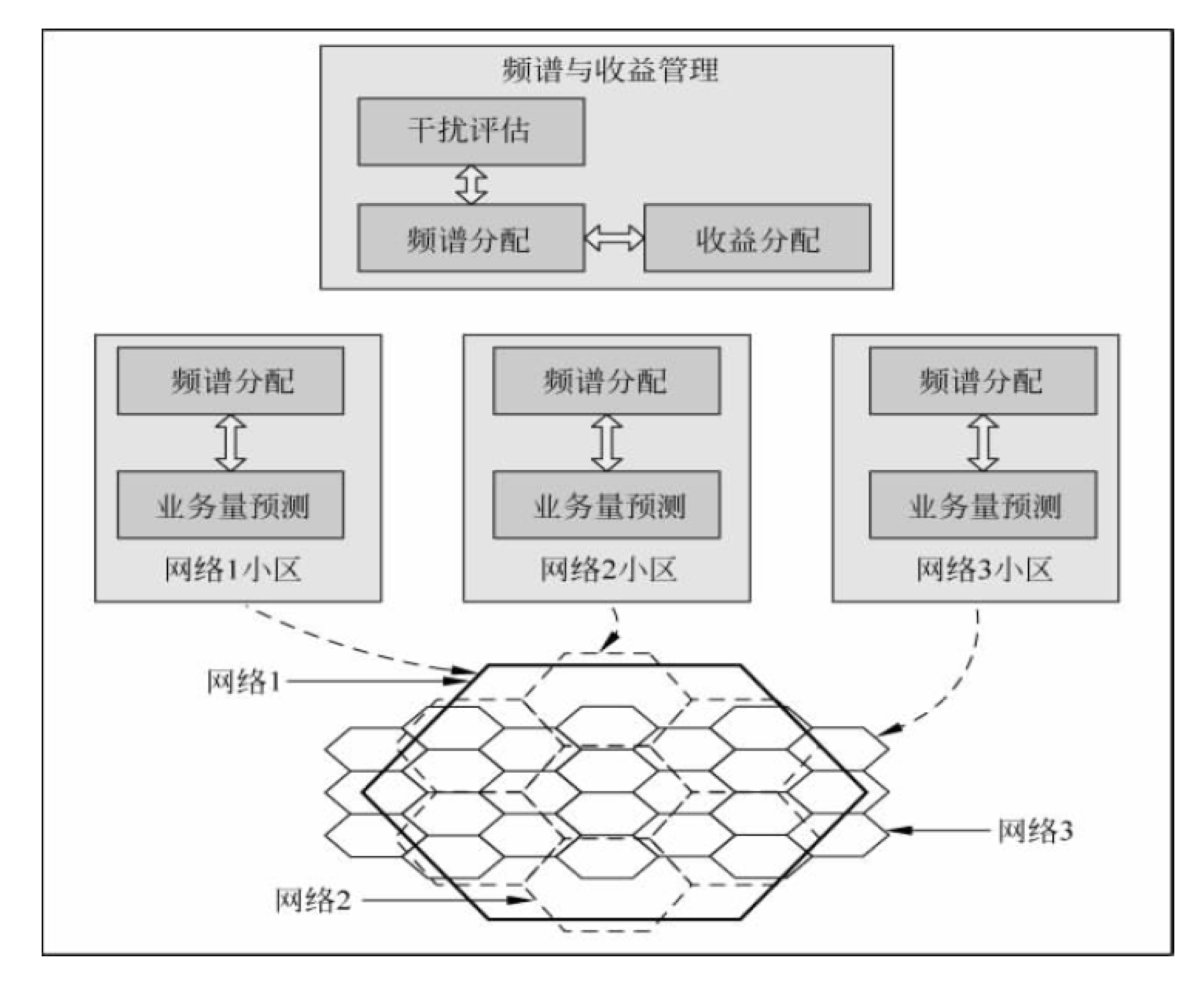


图 7-9　**集中式网络动态频谱管理**

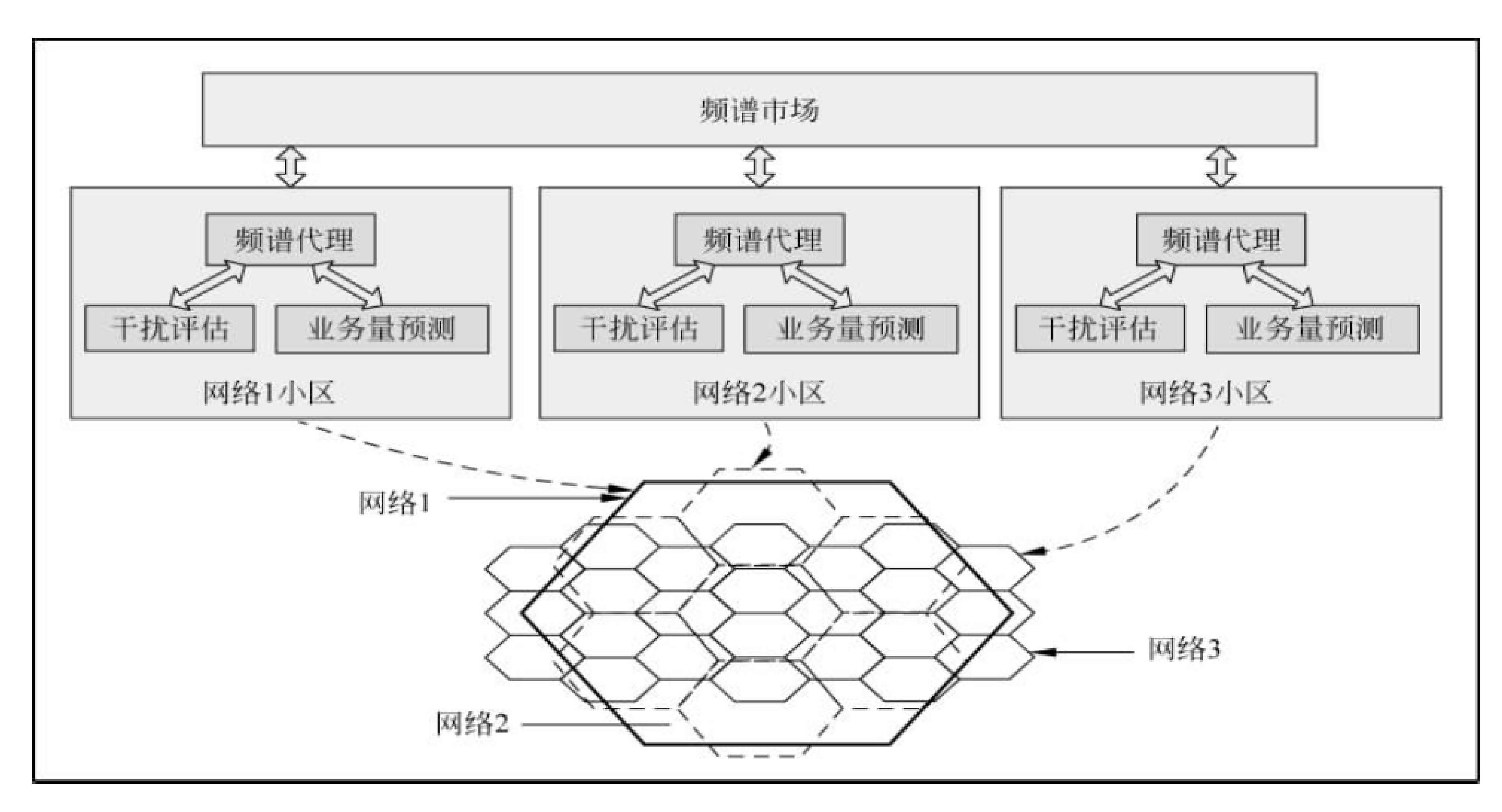


图 7-10　**分布式网络动态频谱管理**

7.3 虚拟网安全问题

在有些机构中，虚拟化已经成为其架构中的重要组成部分。在这里，技术再次走在了最佳的安全方法的前面。随着机构对灾难恢复和业务连续性的重视，特别是在金融界，虚拟环境正变得越来越普遍。我们应该关注这种繁荣背后的隐忧。

虚拟化的计算环境使得其安全问题[9]变得尤为复杂，传统的安全措施很难从根本上解决虚拟化技术的安全问题，因此必须采取新的安全策略。大量的研究实践表明，虚拟化技术的安全问题是多种多样的，主要包括资源运行中的意外情况，如虚拟机溢出、虚拟化服务器宕机；黑客攻击；虚拟机单体安全事件；虚拟机迁移引发的信息泄露；服务器备份中的信息泄露等。这些问题的存在严重影响了虚拟化技术的应用安全。对此，相关技术人员和管理人员应该充分重视起来，采取合理有效的安全策略，切实保证虚拟化技术的应用安全。

1. 底层物理资源安全

物理设备的安全是信息安全的基础和前提。传统的网络结构通过路由器和三层交换机逻辑隔离出一个个相对独立的网络安全域，物理资源在严格策略控制的安全域中可以得到较好的安全保护，网络虚拟化将底层物理资源和服务提供者分开，因而底层物理资源被置于一个相对开放的网络环境中，将面临更大的安全风险。另外，底层物理资源之间的通信数据包与传统网络数据交换有着不同的定义，传统的网络安全设备无法理解这些数据流，也就无法施以正确安全策略。

2. 服务提供商防范黑客攻击

由于服务提供商自身功能的复杂性和开放接口，其设计上难免存有很多漏洞。一般情况下，黑客对于虚拟化服务器的攻击，多是通过应用程序接口或者网络攻击，又或者通过服务器下属虚拟资源进行攻击。对于应用程序接口攻击，应该确保虚拟机只接受经过相应认证和授权的请求；对于网络攻击，应该设置相应的密码保护和防火墙；对于下属虚拟资源的攻击，应该严格限制任何未经许可的用户访问虚拟化软件层，设置必要的安全控制措施。再者，从服务提供商在虚拟网络中所处的核心位置，必然十分吸引攻击者的最大兴趣，成为主要攻击目标，一旦攻破服务提供者，攻击者就会完全占领网络，可向全部网络设备发送指令，为所欲为。例如，将重新定向数据流，造成敏感信息泄露，甚至让整个网络瘫痪。因此，服务提供商一旦出现问题，其产生的后果难以估量。

3. 拓扑变化安全问题

网络虚拟化能够灵活适应业务动态性和快速变化。例如，当用户需要从一个网络拓扑变成另一个网络拓扑时，网络虚拟化管理可以快速调整网络拓扑，在旧物理网络中删除原网络资源，并在新的物理网络中分配现需网络资源。相应地，安全解决方案也应将原有网络设备和安全设备的安全控制（ACL 和 QoS）跟随迁移，然而现有安全产品缺乏对安全策略迁移的支持，导致安全边界不能适应虚拟网络的变化。

4. 用户的安全问题

要提高虚拟化技术的安全性，仅仅依靠以上几种对策还不够，还需要用户提高防范意识，尽量避免涉密或涉及个人隐私的资料存储在服务器中。此外，对登录虚拟化服务的用户进行身份认证，严格验证程序，防止恶意人员的非法入侵。

随着虚拟化技术的快速发展，仅仅依靠服务提供商的自觉性已无法从根本上解决虚拟化的安全问题。除此之外，还应从法律法规角度对虚拟化技术进行安全防护，这就需要国家和政府或相关权威部门制定虚拟化安全环境的运行机制，对服务提供商进行检查和监督，强制要求服务提供商采取必要的安全措施，保证虚拟化技术的安全性。

7.4 网络虚拟化的应用

网络虚拟化是一种重要的网络技术，该技术可在物理网络上虚拟多个相互隔离的虚拟网络，从而使得不同用户之间使用独立的网络资源切片，进而提高网络资源利用率，实现弹性的网络。随着科技的高速发展，虚拟化技术应运而生，而云计算的应用和发展为虚拟化技术提供了新的历史机遇。SDN的出现使得网络虚拟化的实现更加灵活和高效，同时网络虚拟化也成为SDN应用中的重量级应用。

7.4.1 网络虚拟化和云计算的结合

云计算是一种基于互联网的计算方式，可以根据实际需求，将共享的信息资源和软硬件资源提供给相应的计算机和设备，从而将有限的网络资源发挥出最大的效益。云计算实际上可以算是分布式处理、并行处理和网格计算的延续和发展，因此并不是一个新的概念。云计算集成了计算功能和存储功能，能够将服务器、网络、数据库等各种资源，通过互联网的联系，为用户提供相应的服务。云计算具有非常显著的特点，主要包括超大规模、虚拟化、通用性、高可靠性、成本低廉等。

云计算的核心技术就是虚拟化，也可以说，虚拟化是云计算区别于传统计算模式的最为显著的特点。虚拟化技术主要是以软硬件分时服务、模拟与仿真执行等为基础，实现在单个计算机物理设备中，模拟多个相互独立的硬件执行环境为目的。这些虚拟的硬件执行环境也称虚拟机，用户可以在虚拟机上运行不同的操作系统和应用程序。云计算与虚拟化技术存在着非常紧密的联系，虚拟化技术不仅是云计算的核心技术，也是云计算的基础。一个云计算的应用，必然是基于虚拟化的，也只有在虚拟化的环境下，云计算才有可能实现。因此，加强对云计算环境中虚拟化技术的研究，保证虚拟化技术的有效应用，具有非常重要的现实意义和长远意义。

云计算环境的主要特征集中体现在虚拟化、分布式和动态可扩展。在实际应用中，许多的软件和硬件都可以实现各类资源的虚拟化，然后放在云计算平台中进行统一管理。虚拟化技术的应用，打破了物理结构之间的壁垒，也体现出了物理资源向着虚拟资源转变的必然趋势。伴随着计算机技术和互联网技术的快速发展，在可预见的时期内，资源将会实现高度共享，更加透明地运行在各种物理平台上，按照逻辑方式进行管

理，实现资源的自动分配。可以说，虚拟化是云计算的前提，而云计算则是虚拟化的最终目的。

应用虚拟化主要是提供对集中化应用资源的多用户远程访问，将应用变成一种服务，交付给需要的用户。应用虚拟化的基本原理在于，通过对应用程序计算逻辑和显示逻辑的分离，实现界面的抽象化，不需要在用户端安装相应的软件。当用户对虚拟化的应用进行访问时，计算机会自动将相应的人机交互数据传输到服务器端，服务器端根据用户的需求，开设独立的虚拟机，运行应用程序的计算逻辑，同时将处理后的显示逻辑传输回用户端，满足用户的实际需求。应用虚拟化的实现，可以将应用程序从操作系统中解放出来，避免了用户端计算环境对于应用程序的影响，因此具有机动性、灵活性、效率性和安全性等优点。而从用户角度分析，不再需要在计算机上安装大量的应用程序，也不会受到自身计算条件的限制，可以获得极高的服务体验。虚拟的应用程序在使用和操作方面，与原本的应用程序不存在任何差别，用户体验也没有任何变化，因此更容易被广大用户所接受。同时，通过应用虚拟化，可以在同一台计算机上，运行不同版本的应用程序，满足不同用户的使用需求。

7.4.2 网络虚拟化和 SDN 的结合

SDN 不是网络虚拟化，网络虚拟化也不是 SDN。网络虚拟化只是一种网络技术，而基于 SDN 的网络架构可以更容易地实现网络虚拟化。SDN 是一种集中控制的网络架构，可将网络划分为数据层面和控制层面。而网络虚拟化是一种网络技术，可以在物理拓扑上创建虚拟网络。传统的网络虚拟化部署需要手动逐跳部署，其效率低下，人力成本很高。而在数据中心等场景中，为实现快速部署和动态调整，必须使用自动化的业务部署。SDN 的出现给网络虚拟化业务部署提供了新的解决方案。通过集中控制的方式，网络管理员可以通过控制器的 API 来编写程序，从而实现自动化的业务部署，大大缩短业务部署周期，同时也实现随需动态调整。随着 IaaS 的发展，数据中心网络对网络虚拟化技术的需求将会越来越强烈。SDN 出现不久后，SDN 初创公司 Nicira 就开发了网络虚拟化产品 NVP（network virtualization platform）。Nicira 被 VMware 收购之后，VMware 结合 NVP 和自己的产品 vCNS（vCloud Networking and Security），推出了 VMware 的网络虚拟化和安全产品 NSX。NSX 可以为数据中心提供软件定义化的网络虚拟化服务。

目前许多主流的 SDN 控制器，如 OpenDaylight、OpenContrail 等都宣称支持网络虚拟化，像 FlowVisor、FlowN 以及 Advisor 等更是专门为实现网络虚拟化开发的 SDN 组件。利用 SDN 去部署实现网络虚拟化之所以得到越来越多的重视，更重要的原因在于 SDN 优良的结构体系。两层分立的模型使得 SDN 可以实现两种不同级别的网络虚拟化，即数据平面虚拟化和控制平面虚拟化，区别在于后者实现了 SDN 的控制器部分的虚拟化，从而能够提供功能更加强大的 API。目前 SDN 在支持虚拟化方面还存在着一些问题。首先是 SDN 本身的问题，SDN 控制器虽然强大，但是中心化的特征很容易使其成为整个网络的性能“瓶颈”，而且也容易产生安全隐患，如何去中心化（如采用控制器分布式部署或多控制器备份机制）是 SDN 实际部署网络虚拟化首先要完成的任务；其次，

SDN 是基于流的方式实现网络的功能，要实现网路虚拟化中对于网络资源的隔离的需求就要实现流的有效隔离，在这一方面 SDN 还有待加强；最后，在网络虚拟化中，不同的流的服务质量（QoS）要求差异巨大，SDN 需要区分服务，避免时延要求高的流进入控制器部分。

7.5 网络功能虚拟化背景现状及影响

网络功能虚拟化（network functions virtualization，NFV）[10]通过使用 x86 等通用性硬件设备以及虚拟化技术，来承载很多功能的软件处理。网络功能虚拟化是一种对于网络架构的概念，利用虚拟化技术，将网络节点阶层的功能分割成几个功能区块，分别以软件方式实现，不再局限于硬件架构。NFV 将原本专业的网元设备上的网络功能提取出来虚拟化，运行在通用的硬件平台上，通过软硬件解耦及功能抽象，使网络设备功能不再依赖于专用硬件，资源可以充分灵活共享，实现新业务的快速开发和部署，并基于实际业务需求进行自动部署、弹性伸缩、故障隔离和自愈。

7.5.1 网络功能虚拟化背景

传统网络中的各种设备都是基于私有平台部署的，各种网元都是一个个封闭的盒子，各种盒子间的硬件资源无法相互通用，各种设备扩容时需要添加硬件，缩容后硬件资源闲置，耗时长，弹性差，成本高。传统的电信网络架构是在 20 世纪 90 年代以电话业务为主的时代确立的，随着数据业务流量的爆炸式增长，传统网络架构暴露出难以克服的结构性问题，主要体现在以下几个方面：

（1）网络复杂且与业务强相关。每一种新业务的引入都需要新建一张承载网络，而且通常是由功能单一、价格昂贵、专用的硬件设备构成。

（2）网元设备使用软硬件一体化的封闭架构，导致设备日益臃肿、扩展性受限、功耗大、功能提升空间小、技术进步慢、价格昂贵、厂商锁定。

（3）网络和业务相互割裂，缺少协同，业务不了解网络的资源使用状况，网络无法适应业务动态的资源需求，造成资源不能共享、业务难以融合。

（4）成本居高不下，网络中存在大量不同厂商、不同功能的设备，在部署中需要实现多厂商设备的集成、互通、维护和升级，很难降低成本。

现有的相对封闭的网络架构以及粗放的网络建设与运维模式难以支撑网络的可持续发展。新服务针对现有网络提出了降低网络建设与运维成本、提高网络资源利用效率、提升网络与业务部署速度的新需求。

未来的网络将是数字化的、全连接的，云计算、大数据、物联网、移动互联网、工业互联网以及高清视频、虚拟现实等将成为未来的热点业务，运营商处于流量和连接数快速增加、用户体验高要求和新业务不断涌现的状态。在流量方面，运营商网络的流量将会有爆炸式的增长；在连接数方面，预计近些年全球将有千亿数量级物联网终端连入互联网；在用户体验方面，实时互动、按需定制、全时在线、自助服务以及社交分享成为用户的核心需求。新服务提出了对网络敏捷、创新、安全、经济、开放的要求，要求网络遵循开放

标准体系，能够支撑业务多样化、弹性化，支持第三方业务创新，提供高安全性，支持全方位的自动化部署和运行维护。

7.5.2 网络功能虚拟化标准化

传统电信运营商提供的业务是构建在完善的标准基础上。采用 NFV 技术后由于硬件虚拟化技术的应用，传统电信网网络架构发生了实质性的变化。新架构产生了新的逻辑功能单元，如 Orchestrator、VNFM、VIM 等，而这些功能单元与 VNF 及 VM 之间均产生了新的接口。在虚拟化的网络环境下，哪些接口需要进行标准化，哪些接口可以采用私有接口不进行开放，均需要明确制定相关的标准。例如在虚拟化的网络环境下，原有网络的逻辑网元间的接口及协议未发生变化，但原有的接口及网元的交换流程有可能发生变化，在 EMS 与 OSS 之间将新增虚拟化相关的网管信息及设备运行维护相关信息。

ETSI 主导负责 NFV 基础架构，推动 PoC 认证。2011 年 11 月，由运营商主导在 ETSI 成立 NFV ISG 工作组，成为推动 NFV 基础架构标准化的主要国际标准组织之一，主要制定支持 NFV 硬件和软件的基础设施要求和架构规范以及虚拟网络功能的指南。目前已发布架构、需求、应用案例等多个技术文稿及一系列 PoC 文档。目前，该工作组的研究重点包括：IFA 组研究的 MANO 功能及接口、加速技术；REL 组研究的可靠性模型、故障检测及可靠性框架；TST 组研究的测试方法及开源组件等；EVE 组研究的 NFV 网络演进及生态体系建设；SEC 组研究的与安全相关的内容。

3GPP 重点面向移动核心网的演进需求，聚焦网络资源切片。3GPP 中主要由 SA5 负责与 NFV 相关的标准化工作。SA5 侧重于制定虚拟化网络管理架构，云管理与网管协同，NFV 引入后的网络信息模型以及故障、配置、性能、安全等管理流程，OSS/BSS 网管接口要求等方面的标准。

IETF 负责 IP 协议体系的完善工作，构建网络虚拟化能力。IETF 作为互联网领域的重要标准组织之一，也同步开展 NFV 相关标准化工作，涉及两个研究组和 9 个工作组，其中 NFV RG 主要关注固定和移动网络基础设施的虚拟化、基于虚拟化网络功能的新网络架构、家庭和企业网络环境的虚拟化、虚拟化和非虚拟化基础设施与服务的并存等问题研究；SDN RG 主要针对 SDN 模型进行定义和分类，网络描述语言(和相关的工具)，抽象和接口，网络或节点功能的正确操作验证等。IETF 的 9 个工作组涉及 Internet、路由、传输、安全 4 个领域，包括 DMM、SFC、NVO3、I2RS、BESS、TEAS、VNFPOOL、IPPM、I2NSF 等，研究内容涵盖移动网络，数据中心内部网络虚拟化，用于网络安全控制和监控功能的新信息模型、软件接口和数据模型等。

OPNFV 倡导开源和集成，是 NFV 开放平台项目，由 AT&T、中国移动等电信运营商牵头发起的开源组织，于 2014 年 9 月 30 在 Linux 基金会下创建成立，该开源社区旨在提供运营商级的综合开源平台以加速新产品和服务的引入，实现由 ETSI 规定的 NFV 架构与接口，提供运营商级的高可靠、高性能、高可用的开源 NFV 平台。OPNFV 项目启动以来，已经得到 100 多个厂商关注，包括网络运营商、IT 厂商、设备制造商及解决方案提供商等。

7.5.3 网络功能虚拟化的意义

网络功能虚拟化NFV技术产业生态日益完善，产业重心加速调整。与传统软硬件封闭一体的架构相比，采用NFV技术将推动硬件和软件解耦，以及软件功能的分层解耦，进一步细化和拉长产业链环节，形成新的NFV产业生态要素。在这种新的产业生态体系下，产业重心将由硬向软调整，软件公司特别是第三方系统集成商的产业价值凸显。传统的由单一厂商提供整套软硬件一体的系统，将分解成来自不同厂商的组件，如通用硬件、虚拟化平台、不同功能的VNF等。这一方面要求网络架构更加开放，接口统一并且标准化，另一方面要求第三方系统服务集成商具备更强的软件研发及系统集成能力。对于传统通信设备行业巨头来说，则需要适应这种产业生态的变革，围绕自身平台积极打造产业生态系统，提供开放架构的产品及解决方案，集成源自不同服务提供商的组件。

网络功能虚拟化将深刻改变基础网络运营商的网络建设、运行维护、业务创新和管理模式。在网络建设方面，NFV利用通用化硬件构建统一的资源池，不仅能够大幅降低硬件成本，还可以实现动态按需分配网络资源，达到资源共享和资源利用率显著提升的目的。在研发和运行维护方面，NFV采用自动化集中管理模式，这将推动实现硬件单元管理自动化、应用生命周期管理自动化以及网络运行维护自动化，运维研发一体化(DevOps)成为可能。在业务创新方面，基于NFV架构的网络中的业务部署只需申请云化资源(计算/存储/网络)，加载相关软件即可，这些使得网络部署和业务创新更加简单。在企业管理方面，为了适应NFV给运营商带来的一系列变化，基础网络运营商的管理模式、组织关系、企业文化等都需要变革，运营商的企业文化将加速向软件文化转变。

网络功能虚拟化正在推动通信设备制造业的发展重心调整、产业升级与生态重构。NFV拉长了整个通信产业链条，传统设备制造商面临严峻挑战。引入NFV以前，旧有产业链相对单一，核心成员主要包括设备制造商、芯片制造商等，而NFV引入后，新的产业链核心成员主要包括通用硬件设备制造商、芯片制造商、虚拟化软件提供商、网元功能软件提供商、管理设备提供商等。其中受冲击最大的是传统设备制造商，原本软硬件一体化设备销售模式被拆解为通用硬件、虚拟化平台和网元功能软件三部分的销售模式，传统设备制造商除了在网元功能软件上具有强的技术壁垒外，在通用硬件和虚拟化平台软件方面将面临来自IT领域的强大竞争。

网络功能虚拟化将极大激发互联网企业与第三方业务服务商的业务创新活力。NFV软件化、模块化实现方式可灵活地对网络功能进行定义、组合和管理，对外提供更为丰富的网络功能接口，促进第三方业务服务商与运营商的灵活对接，实现业务的快速集成和上线，激发第三方业务创新活力，互联网企业将不断推出相关创新型产品。

网络功能虚拟化的应用会为网络运营商带来诸多效益，也会为电信通信行业带来巨大的变革。通过整合设备以及利用IT行业的规模经济效应，来降低设备成本和能源消耗。缩短传统运营商的创新周期，加快了业务推向市场的速度。基于软件的开发部署模式使得功能演进的模式成为可能，考虑基于硬件模式的那种投资经济性的限制也不再适用。网络功能虚拟化能够使得网络运营商显著地减少业务的成熟运行周期，加速业务上线速度，提高创新能力。在统一的基础架构上运行业务，测试业务，将有力保障了有效的

测试和集成工作，减少开发成本，加快业务上线速度。基于地理位置或者客户群的针对性服务成为可能，能够根据具体需求快速地扩展或者降低服务能力。另外，基于软件的远程服务无须增加新的硬件设备，这也有助于服务及时性的提升。促进形成一个广泛多样的产业链，并鼓励开放。可以开放虚拟一体设备市场给单纯的软件商、自由开发者或者研究人员，鼓励这些开发者带来更多的创新，在很低的风险下带来新的服务以及新的收入。基于实际的数据流量、移动用户以及服务需求，在线实时地优化网络和技术。例如，对网络功能自动和在线实时地调整优化响应位置及资源配置，可以提前避免出现系统故障，因此，无须准备设备以及人员的冗余备用。多租户的支持能容许网络运营商为不同的用户应用以及内部系统和其他运营商提供可定制化的服务和互通业务，在同一硬件平台上的共存业务将都具有合适的安全分离的管理域。利用标准主机和存储的电源管理功能，以及工作负载的整合和位置优化等特点来降低能源消耗。例如利用虚拟化技术，可以在业务非峰值期（例如晚上）将工作负载集中在很少量的几台机器上，而其他的机器可以关闭或者进入电源节能模式。物理网络的统一标准，以及其他所支持平台的统一，这些所带来的好处是能够提高运维效率。运维标准化的 IT 高容量服务器的产业规模远大于当前的电信专有网络市场规模，也更为完整统一，这有助于避免对基于特殊设备的应用需求。

7.6 网络功能虚拟化架构

相比于当前的网络架构（独立的业务网络＋OSS 系统），网络功能虚拟化从纵向和横向上进行了解构，如图 7-11 所示[11]。其中，OSS 表示开放存储服务；BSS 表示基站子系统；NFV 表示网络功能虚拟化；EMS 表示网元管理系统；VNF 表示虚拟网络功能；NFVI 表示网络功能虚拟化基础设施。根据 ETSI 标准化组织提出的架构，纵向将 NFV 网络分为三层：基础设施层、虚拟网络层和运营支撑层。横向分为业务网络域和管理编排域。通过这样的组织架构，一个业务网络可以通过 MANO 来自顶向下分解，直到可分配的资源，然后对应 VM 等资源由 NFVI 来分配，对应 VNFL 资源需要同承载网管理系统交互，由 IP 承载网来分配。

1. 基础设施层

NFVI 是一个资源池，包括 IT 资源和 CT 资源（通信网的传输资源和交换路由资源等）。NFVI 包括虚拟化的计算、存储、I/O 资源池和用于 NFV 通信网的传输资源、交换路由资源等，为 VNFs 提供部署、管理和执行的硬件/软件环境。硬件资源是指通过虚拟化层为 VNFs 提供处理、存储和连接能力的计算、存储和网络资源。虚拟化层对硬件资源进行了抽象，并把 VNF 的软件功能和底层的硬件解耦合，从而保证硬件资源与 VNFs 的无关性。

2. 虚拟网络层

虚拟网络功能（VNFs）是传统电信网络设备在网络功能虚拟化系统中的展现形式，VNF 可部署在一个或多个虚拟机（VM）上，满足电信系统的功能需求，VNF 所提供的网

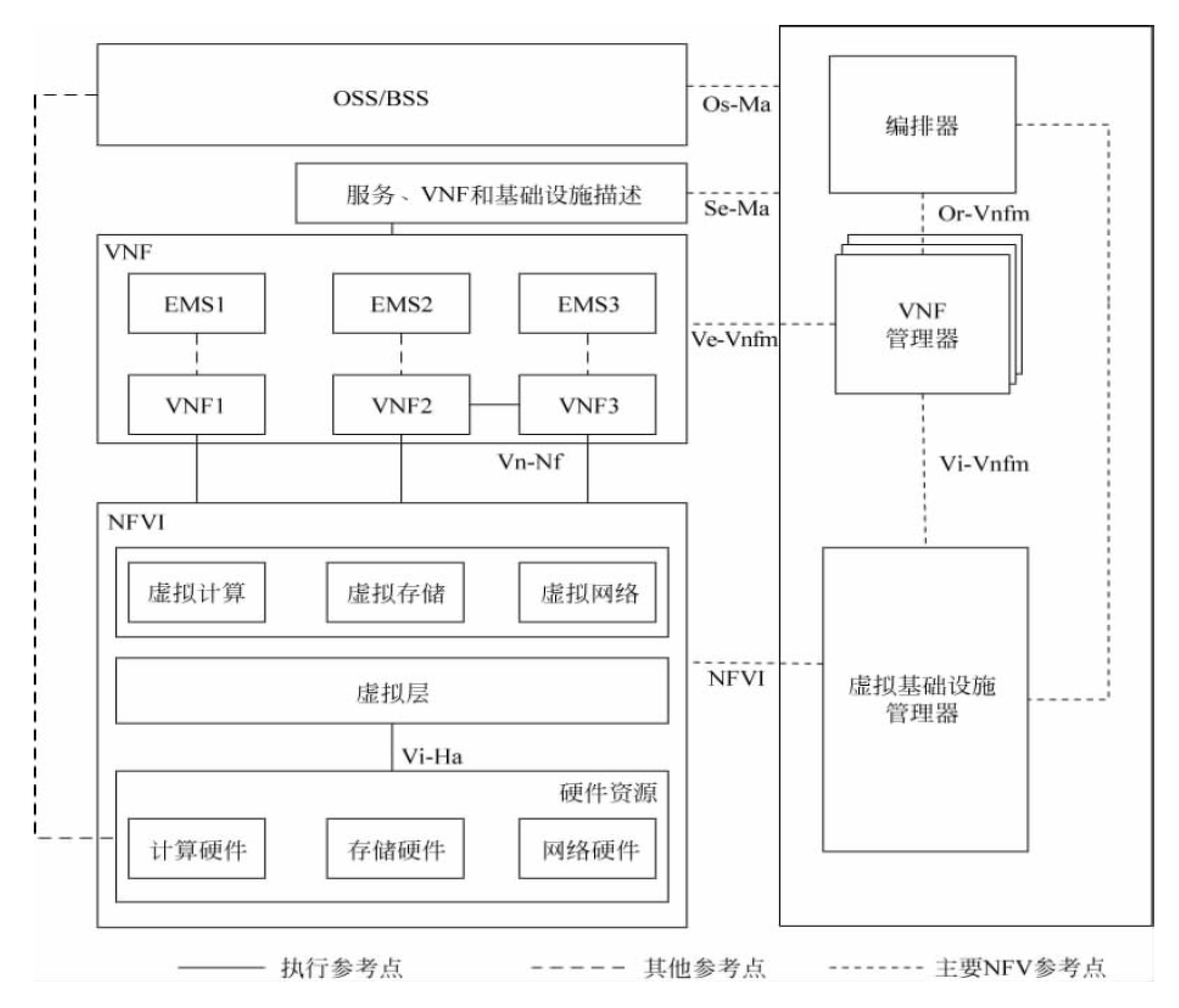


图 7-11　网络功能虚拟化架构

元功能与非虚拟化时的网元功能应保持一致，与其他网元实体的接口与非虚拟化时的接口应保持一致。网元管理系统（EMS）负责完成传统的网元管理功能及虚拟化环境下的新增功能。EMS 通过北向接口与网管系统相连，提供配置管理、警告管理和性能管理等功能。

3. 运营支撑层

运营支撑层是指目前的开放存储服务（OSS）/基站子系统（BSS），但需要针对网络功能虚拟化进一步修改和调整。OSS/BSS 网元在支持传统网络管理功能基础上，还需要支持在虚拟化环境下与编排器 Orchestrator 交互，完成维护与管理功能。

4. 管理编排域

NFV 和传统网络明显区别就在于增加了管理编排域（management and orchestration，MANO）[12]，MANO 定义了提供虚拟网络功能 VNF 的框架，并可以对此进行配置和操作，本质上是定义了一个能够控制所有 VNF 的编排器，同时具有应用程序北向接口和组

件南向接口。MANO 负责整个 NFVI 资源的管理和编排、业务网络和 NFVI 资源的映射关联、OSS 业务资源流程的实施等。MANO 内部包括虚拟基础设施管理器(VIM)、虚拟网络功能管理器(VNFM)和编排器(orchestrator)三个部分,分别完成对 NFVI、VNF 和 NS(network service,业务网络提供的网络服务)三个层次的管理。VIM 负责虚拟化基础设施管理,主要管理监控整个基础设施层资源,包括硬件资源和虚拟资源。VNFM 负责管理 VNF 单元,VNF 单元的生命周期管理,包括实例化、删除、查询、改变容量、终结等,VNFM 应提供基于业务容量模型的 VNF 自动部署和手动部署能力,能够自动或手动完成 VNF 的实例化。VNFM 支持 VNF 软件包管理,VNF 软件包包括 VNFD、GUEST OS 镜像文件以及 VNF 软件镜像文件。VNF 包管理包括 VNF 包的加载、更新和删除。VNFM 应根据 VNF 的资源利用情况,发起扩容/缩容操作,负责 VNF 所用虚拟资源的性能数据/事件的采集和 VNF 业务所使用虚拟机故障信息的采集。编排器负责提供硬件资源和虚拟资源的视图,监控硬件资源和虚拟资源,完成性能统计和故障管理工作,控制 VNFM 管理 VNF 软件包,以及 VNF 实例的创建、更新、终止和弹性伸缩。同时还需要提供管理接口供操作员进行云管理系统的本地维护,支持网络服务的加载、部署,与 OSS 协同管理网络服务。

7.7 网络功能虚拟化关键问题

网络功能虚拟化改变了传统电信设备的设备形态,实现了硬件和软件的完全解耦,这对网络运营商的运维和管理体系带来了新的挑战。基于网络功能虚拟化(NFV)的核心网需要解决一些关键问题,才能实际进行部署。与传统网络一样,网络功能虚拟化需要考虑管理、运营与维护问题,涉及资源调度与协调编排问题、可靠性与安全性问题、部署与互联问题及运行维护方式的转变问题等。

1. 资源调度与编排

在网络规划与部署方面,虚拟化网络与传统电信网存在很大差异。目前传统的核心网直接依据业务量线下测算网络容量、确定网络拓扑及网元数量进行规划。虚拟化的核心网则自动依据业务量通过编排器进行 VNF 规划,并落实到 VM 的规划。虚拟化架构中业务编排模式、架构的管理、协调层中各个逻辑功能及其协调工作机制都需要研究。虚拟化核心网中底层硬件资源、虚拟化网元功能、协调管理部分不再局限于同一个厂商。在网络集成等方面也需要进行研究,形成新的虚拟化核心网业务部署及资源调度、机制编排。

NFV 要实现硬件资源与软件功能的解耦,需要通过标准的接口、通用的信息模型以及信息模型与数据模型之间的映射来实现,同时还需要一套新的管理和编排功能系统。NFV 引入了管理和编排功能(MANO),主要用于提供虚拟化资源、虚拟化网络功能和网络业务的统一管理,包含三类功能模块:NFV 编排器(NFVO)、VNF 管理器(VNFM)和虚拟化基础设施管理器(VIM)。MANO 负责业务编排、虚拟资源需求计算及申请,完成网络功能部署,打破现有业务部署的流程格局,对现有的设备采购模式和运维模式都会产生较大的冲击。

VNF 生命周期管理是 NFV 架构下实现自动化运维的关键环节，由 MANO 和 EMS 协同完成，包括 VNF 的实例化、终结、查询、扩容/缩容、自愈等功能，实现了自动化资源编排、智能部署编排、弹性扩缩容决策等。在具体部署时需根据业务的规格需求和基础设施的硬件属性完成自动化资源编排，根据亲和性/反亲和性策略、基础设施资源的负载/可用状况等关键要素完成智能部署编排，基于网元的 CPU 占用率、用户容量门限、带宽使用率等关键要素进行扩缩容决策。上述算法的参考要素和具体实现需要业界共同研究。

2. 可靠性与安全性

NFV 要求的硬件通用化、网元功能的软件化导致网络 I/O 能力无法满足电信网络的需求，计算能力无法满足特殊功能（如加解密、编解码、深度报文解析等）的需求。同时 NFV 引入中间件带来一定的性能损耗。NFV 技术需要降低软件的开销、引入软硬件加速技术以满足电信网络高速转发、密集计算的性能需求。

虚拟化核心网的性能问题主要体现在内存读写上。NFV 增加了虚拟网络层，虚拟化核心网的性能也因此受到影响，特别是对延迟敏感并有性能要求的用户方面。虚拟化核心网的性能下降主要是由内存输入、输出端引起的。虚拟模块主要为 IT 环境搭建，因此对于电信应用中更多的网络和数据业务流意味着更多的缓存需求。目前业界采用 DPDK、SR-IOV 等手段提升虚拟化核心网用户方面的性能指标，也有设备提供商采用特定设计内存和 IO 读写算法来提升性能指标。基于服务器的虚拟化核心网是否能满足大量数据业务、语音业务的用户方面媒体转发的性能指标需求，也需要实践进一步考查。

虚拟化的核心网硬件设备基于通用服务器。通用服务器与传统的电信硬件相比在可靠性方面有所下降，但随着通用服务器的发展，其可靠性也在日益提升[13]。另外，在虚拟化的架构下，除了虚拟网元间依旧保持原冗余备份外，虚拟化的网络资源可以实现分布式计算及分布式存储。当底层资源故障时，虚拟化网络可自动将计算任务转移到其他 VM，或者从其他单元自动调用备份数据，从而提升系统的可靠性。同时，虚拟化的控制层可更大范围地调用硬件资源，在异地容灾等一系列领域中提升系统的可靠性。但 IT 云的可靠性算法不能达到电信云的可靠性指标，需要通过新的虚拟化环境下的可靠性算法及现网测试才能不断落实虚拟化网的可靠性问题。传统的电信网络设备需要 99.999%的高可靠性，而采用虚拟化技术的通用设备目前暂时还没达到该要求（一般 COTS 设备可靠性只能达到 99.9%）。为了满足电信级设备高可靠性的要求，NFV 的组件需在多个方面提供与传统电信设备相当的性能，包括故障率、检测时间、恢复时间以及检测和恢复的成功率。

相比传统电信设备，软硬件分离的特点以及虚拟化网络的开放性给网络带来了新的潜在安全问题：一是引入新的高危区域虚拟化管理层；二是弹性、虚拟网络使安全边界模糊，安全策略难于随网络调整而实时、动态迁移；三是用户失去对资源的完全控制以及多租户共享计算资源带来的数据泄漏与攻击风险。在 NFV 环境中，可能存在安全风险的关键组件包括 VNF 组件实例、绑定到 VNF 组件实例的本地网络资源、远程设备上对本地 VNF 组件实例的参考、VNF 组件实例占用的本地、远程以及交换存储等。在发生安全事故的情况下，如何保证这些关键组件所涉及的硬件、内存不被非法访问，如何保证

VNF 上应用的现有授权不被改变，本地和远程资源彻底清除崩溃的 VNF 资源及授权不被滥用，是 NFV 安全面临的关键技术挑战。针对安全性问题，目前采用的解决方案是通过多级认证，保障资源使用的合法性，防止盗用及恶意接入。同时，采用虚拟资源隔离的方式避免虚拟应用间的互相影响，也可隔离不同用户，为政企客户等分配相对独立的资源及管理，从而提升安全性。

3. 运行与维护

传统电信设备的故障定位在单台设备范围内进行，厂商为设备的性能、可靠性、业务指标负责，故障定位简单。虚拟化之后，故障定位将涉及虚拟功能网元、虚拟层、硬件层，尽管针对不同的技术路线会有所区别，但故障定位都不再局限于一台物理设备范围之内，对运维人员的技术能力提出了更高的要求。传统设备的网络管理系统和资源管理系统通常合并在一起，由 OSS 统一管理，而在虚拟化之后，设备将运行在虚拟机上，网管系统将不能直接看到硬件设备的信息，网管系统是否需要修改，该如何修改，硬件与虚拟网元功能解耦之后，硬件可以统一管理，但如何统一管理，归属传统网管系统，还是和数据中心的基础设施统一管理，这都是有待结合运营商的具体网络情况详细分析，制定相关策略。在网络规划和建设方面，运营商通常根据网络业务的发展情况，采购对应的网络设备提供业务；虚拟化之后，硬件资源、虚拟资源和虚拟功能网元解耦，硬件资源、虚拟资源可以预先统一采购和建设，业务需要时，直接从现有资源中调配相应的资源供虚拟功能网元使用，便于业务的快速部署和建设，但这将对运营商的现有采购流程和采购模式产生影响，因此运营商还需要探索新的采购流程和采购模式。

4. 集成部署与互联

NFV 本身解决的是业务网络的自动部署问题，从架构看也是一个巨大的 ICT 系统集成工程，包括 NFVI 的集成、VNF 的集成和业务网络的集成，涉及的系统、厂商、地域、接口都非常多。引入 NFV 后，电信网络从传统一个厂商完成的软硬件集成转换为 NFV 下多个厂商的软硬件集成，复杂度大大提升。现阶段，NFV 相关接口的标准化进度不一，部分接口将直接采用开源软件，部分 API 难以完全标准化。此外，开源软件和厂商定制化软件解决方案所采用的私有协议和接口都将成为 NFV 系统集成和工程联调面临的挑战。

NFV 通过标准 API 接口向第三方应用、中间软件供应商等开放电信网络能力，可以通过统一的基础服务平台快速集成第三方业务。当前，如何规范开放的 API 编程接口，如何基于敏捷开发方法构建统一的基础服务平台，从而动态适配不同的第三方应用是 NFV 发展面临的挑战。

7.8 网络功能虚拟化的应用场景

ETSI NFV ISG 在定义网络功能虚拟化的同时，也梳理了网络功能虚拟化的重点应用场景[14]，如图 7-12 所示。其包括 NFV 基础设施即服务（network functions virtualization infrastructure as a service，NFVIaaS），允许服务提供商在提供给终端用户的独立管理的 NFV 基础设施（NFVI）上进行服务的提供保证及收费；VNF 软件即服务（virtual

network function as a service，VNFaaS），VNF 通常被认为在网络运营商的私有云模型上执行，VNFaaS 能够提供远程网络功能；虚拟网络平台即服务（virtual network platform as a service，VNPaaS），VNPaaS 模型提供给用户更宽泛的控制，用户能够进行多个 VNF 实例的配置；VNF 转发图（VNF forwarding graphs，VNFFG），VNFFG 在 VNF 之间提供了逻辑连接，可定义数据包的传输路径；移动核心网的虚拟化（virtualization of mobile core network）；IP 多媒体子系统虚拟化（virtualization of IP multimedia subsystem，vIMS）；移动通信基站的虚拟化（virtualization of mobile base station）；家庭环境的虚拟化（virtualization of the home environment）；CDN 的虚拟化（virtualization of content delivery network，vCDN）；固定接入网络功能的虚拟化（fixed access network functions virtualization）。下面详细介绍其中几个典型的应用。

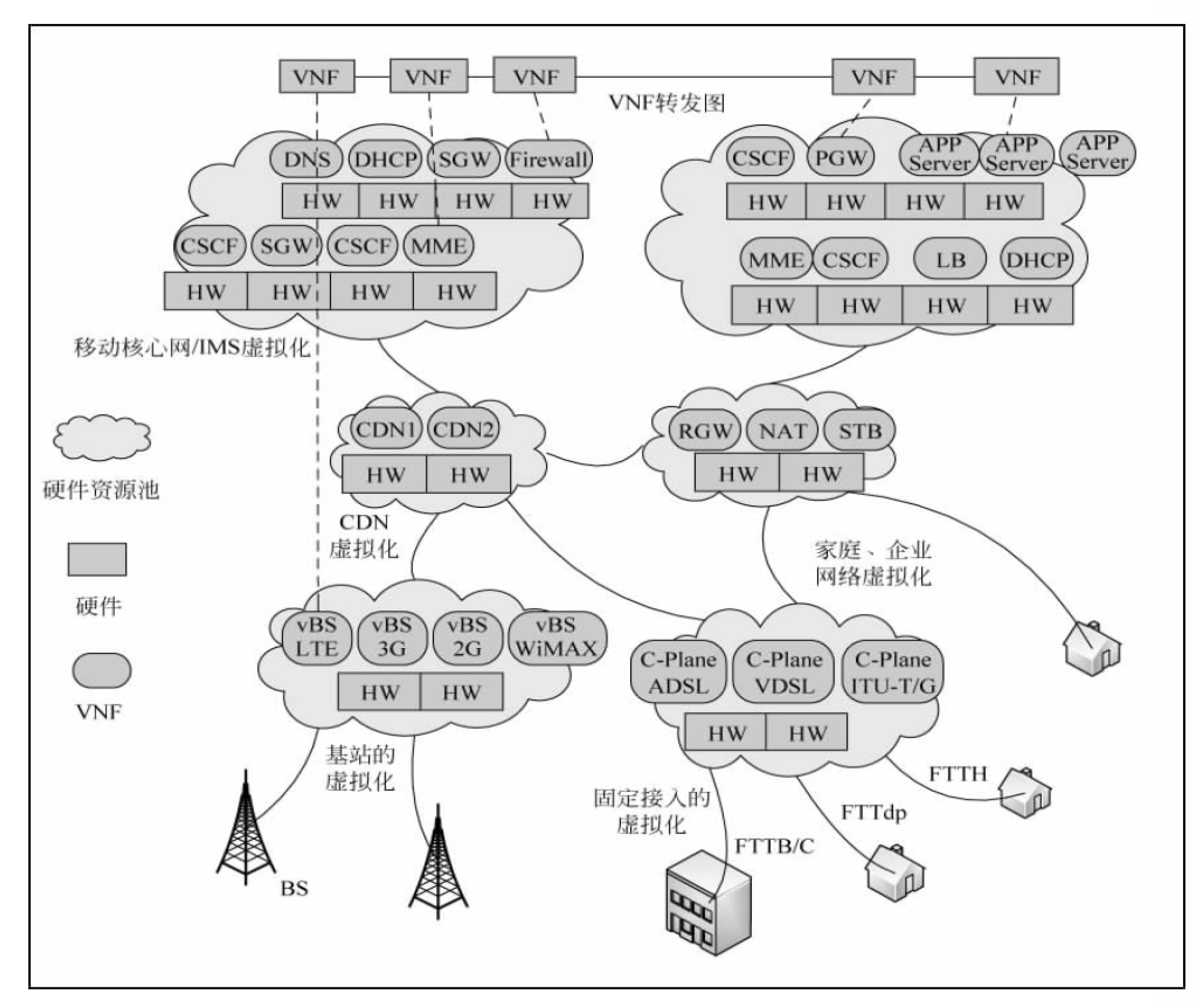


图 7-12　网络功能虚拟化应用场景

1. 移动核心网虚拟化

移动核心网通常包括电路域（CS）、分组域（PS）、IMS 域三类。电路域主要承载传统

的语音业务，是2G、3G网络时代的主要业务支撑实体，但是随着4G业务的发展，传统的语音业务将逐步萎缩或者最终退出服务。移动网络的分组网在2G、3G时代由于无线接入能力有限；在4G时代，由于无线接入能力大幅提升，将面临新的发展机遇。IMS业务是传统语音业务的升级，融合通信可以大幅提升传统电信业务的用户体验，IMS域同样面临着发展机遇。移动核心网的虚拟化能够降低网络复杂性，减少费用，并提供更高的网络使用率及服务质量，主要虚拟化目标是EPC网络功能和IMS网络功能的虚拟化。

IP多媒体子系统虚拟化(vIMS)网络可以快速调配硬件资源池中的资源，可以快速搭建业务测试环境，可以对预上线的业务进行上线测试，将有助于运营商缩短业务上线时间，提升市场的竞争能力。

传统EPC(evolved packet core)设备为专用的硬件设备(大多数为ATCA设备)，设备通用性差导致研发、测试、入网和运维周期长，并且成本难以下降。核心网EPC网元虚拟化vEPC[15]采用“应用+控制器+转换”的三层架构，将网元的流量流向、具体流量处理等控制功能提取出来并由“应用+控制器”两层来实现，“转换”层实现基于流的转发功能，并逐步实现控制面网元的集中化。同时，通过将SAE网关的信令面与MME、PCRF等设备相融合，形成移动核心网虚拟控制云。EPC网元虚拟化通过NFV实现网络硬件架构的统一，解决容量增加带来的成本问题，通过业务控制和转发的分离以及硬件和软件的分离解决业务灵活部署和增强的问题，从而降低运营商的CAPEX和OPEX。vEPC通过通用硬件构建虚拟化的统一平台，支撑EPC网元(包括ME、HSS、PCRF、SGW、PGW)的高效部署，从而降低建网和运行维护的成本。引入虚拟化后，vEPC网络架构、接口及协议依然遵循原有的标准规范。

2. 家庭网络虚拟化

家庭网络虚拟化[16]是将家庭网络中的家庭网关(HG)、机顶盒(STB)设备中的控制面功能及业务处理功能(如防火墙、地址管理、设备管理、故障诊断等)分离出来，虚拟化后迁移到控制器侧或云端，HG及STB设备上仅保留物理接入接口(广域网口、局域网口、USB接口等)以及数据面二层转发。家庭网络虚拟化一方面可以简化用户面设备，运营商不需要对STB和HG进行持续的维护和升级，只需通过远程方式即可为用户提供网络故障诊断服务，便于故障诊断和修复，提升业务可管理性并降低能耗；另一方面可以提高业务部署的灵活性，从而可以为未来新业务的快速部署提供能力，缩短了新业务市场的响应时间。

3. 内容分发网络虚拟化

内容分发网络(CDN)通过在网络各处放置节点服务器所构成的在现有的互联网基础之上的一层智能虚拟网络，CDN系统能够实时地根据网络流量和各节点的连接、负载状况以及到用户的距离和响应时间等综合信息将用户的请求重新导向离用户最近的服务器节点上。其目的是使用户可就近取得所需内容，解决Internet网络拥挤的状况，提高用户访问网站的响应速度。CDN的虚拟化(vCDN)目标是使得内容分发服务的扩展变得更为容易，也使得通过按需安装更多的其他服务分发程序(如Web加速)来完成最大化硬件重利用的目标。CDN的虚拟化也允许从潜在的商业合作伙伴中获取CDN服

务，如一些外部的 CDN 服务提供商。现今的 CDN 缓存节点通常是专用物理设备，这会带来很多弊端，虚拟化的目标是 CDN 的所有组件，但会优先考虑 CDN 缓存节点。

4. 移动通信基站虚拟化

移动通信基站在一定的无线电覆盖区中，通过移动通信交换中心，与移动电话终端之间进行信息传递。移动通信基站虚拟化将部分的无线接入网(RAN)节点虚拟化，采用标准 IT 硬件设备，主要虚拟化目标是传统 RAN 节点，能够获得更低的能源消耗，更便捷地管理操作，更快地投入市场。

5. 数据中心网络虚拟化

数据中心网络虚拟化[17]可通过 Overlay 方式全面屏蔽底层物理网络设施，以软件方式实现底层物理网络的共享和租户隔离，实现针对每个租户的单独网络定义(组网、流量控制、安全管理等)，云数据中心资源管理平台通过 API 接入 SDN 控制器，并通过可编程方式实现多租户网络的灵活部署(包括跨数据中心部署)。数据中心网络虚拟化方案无须依赖底层网络，可以灵活实现不同租户的安全、流量、性能等策略，实现多租户模式，基于可编程能力实现网络自动配置。但是引入 Overlay 后可能会使网络架构复杂化，并且物理网络无法感知逻辑网络，而通过软件控制逻辑网络也会对网络性能产生一定影响。

6. 客户终端设备虚拟化

传统客户终端设备(CPE)在定制化家庭网关/企业网关应用中存在提供新业务能力差、升级周期长、三层配置复杂且故障率较高、网络演进困难等诸多问题。vCPE 是将传统 CPE 上的三层路由、网络翻译(NAT)、用户认证、组播控制、增值业务等功能上移到网络侧，客户端设备仅保留二层转发、L2TP 隧道封装及配置、基于二层信息的防火墙等功能。该方式简化了客户终端设备的配置难度，从而降低用户面故障率，避免对网关频繁升级引起的故障以及硬件、软件成本增加，有利于网络演进。目前，vCPE 已在部分运营商网络中进行试点，产业链相对较为成熟。

7. 宽带远程接入服务器虚拟化

智能边缘是城域网的关键节点，是用户接入的终结点及基础服务的提供点。专业一体化设备在业务功能实现上与硬件强相关，给新业务部署带来很大难题。宽带远程接入服务器虚拟化(virtualization of broadband remote access server，vBRAS)是实现智能边缘虚拟化的代表技术，其以功能集为单元对设备控制平面进行重构，形成用户管理、组播、QoS 与路由等独立模块，每个模块可按需在虚拟机上部署，且可基于通用服务器的虚拟化资源提供能力实现灵活扩展。

8. 路由器虚拟化

为了实现虚拟私有云与企业租户的内部网络互通，需要通过虚拟私有云网关在虚拟私有云与企业内部网络之间建立 VPN。路由器虚拟化(virtualization of service router，vSR)运行在标准的服务器上，可提供路由、防火墙、虚拟专用网(VPN)、服务质量(QoS)

等功能，帮助企业建立安全、统一、可扩展的智能分支，精简分支基础设施的数量和投入。目前，世界上主要的几大公有云服务器提供商，包括亚马逊、谷歌、微软，国内的阿里云、腾讯云等，都在虚拟私有云(virtual private cloud，VPC)内部出口处，提供 VPN 网关业务。随着 VPC 应用的增加，vSR 的应用将越来越广泛。

7.9 电信网络与 NFV

电信运营商为了向网络功能虚拟化方向演进，必须考虑数据中心的设计建设问题以满足 NFV 的基础需求，同时电信业务种类繁多，有的强调实时性，有的必须满足可靠性，也要求响应国家绿色节能的号召，这都是对 NFV 发展方向提出的需求。NFV 网络部署不能一蹴而就，需要提出合理的传统电信网络向 NFV 网络的演进策略。

7.9.1 数据中心建设

数据中心要考虑分级理念，每一级负责所属的覆盖范围，在建设过程中要具体问题具体分析，针对不同的要求和不同的环境背景综合协调以期达到建设需求。

1. 数据中心分级原则

电信业务数据中心的建设需要根据用户和业务发展分布的具体情况来规划，可分为全国(中心/核心)、省级(区域)、接入三级数据中心。全国(中心/核心)数据中心主要承载集约化运营业务，如全国范围内的集中性业务 RCS(融合通信)等，主要涉及信令控制方面的网元。省级(区域)数据中心承载有属地化需求的业务和需要区域集中流量的网元，如 IMS 的控制面网元、SAE(系统架构演进)的 MME 和 SAE-GW 网元等。接入数据中心部署对转发极度敏感的网元(时延要求极高)，如用户面网元和接入网设备，可满足大量业务的本地分流和实施业务就近接入，满足用户业务的体验诉求。

2. 数据中心的覆盖范围

全国(中心/核心)数据中心按照国家或者洲一级的区域进行划分，规划一个或者多个，覆盖全球或全国；省级(区域)数据中心按照省一级的区域划分，也可以把多个相邻的省份划分到一个大区数据中心里面，覆盖半径在千公里级别；接入数据中心按照市县一级的区域根据业务的收敛半径划分，因业务需求不同覆盖半径可以在百公里级或十公里级别。

3. 数据中心的建设

对于大中型数据中心(DC)机房，如全国(中心/核心)DC、省级(区域)DC、大中型本地 DC，侧重于机房的建设与维护成本，尤其需关注长期运维费用的降低。在建设方面，需重点考虑靠近资源、设备低功耗、高密度与集中管理等因素，同时机房选址还需要考虑容灾备份对承载网的要求，如承载网的网络拓扑、网络质量，不同 DC 之间交互信息的带宽和时延等。对于小型数据中心(DC)机房，如小型本地 DC、边缘 DC，则更关注于机房

改造成本，尤其是原有 PSTN 机房在改造过程中面临的电力、空调等问题。

数据中心内采购的设备（包括计算、存储、网络等）必须符合电信级应用环境的要求，对于构建 NFV 基础设施的设备可靠性要求更高一些。

对于电信业务，建议按管理域、业务域和 DMZ（demilitarized zone，不可信的外部网络和可信的内部网络之间的缓冲区）域三类功能域进行规划。管理域主机部署管理软件及网络业务的 EMS/VNFM 软件等；业务域主机部署网络业务的核心业务网元软件等；DMZ 域主机部署网络业务，对外直接提供接入或访问服务的网元软件，如 IPsec VPN 网关。

在业务域内，不同的业务可以划分不同的 VPC（虚拟私有云），实现不同业务的逻辑资源隔离，避免不同业务之间相互影响或资源抢占。

7.9.2 电信网络对 NFV 的需求

电信运营商为了部署 NFV，实现电信网络向 NFV 方向的逐步演进，要求 NFV 具备一定的计算处理能力、网络安全性和可靠性以及充分的信息转发能力，同时还要满足组网质量要求，考虑环境因素如温度、湿度的影响，达到一定的能耗标准[18]。

1. 计算处理能力

与传统电信设备相比，NFV 在物理硬件之上增加了虚拟化层，这会导致物理资源的损耗，需要采用性能提升技术来降低该损耗的影响。

（1）CPU 处理能力。多虚拟机对物理 CPU 资源的竞争和抢占，导致虚拟机的性能降低，尤其是转发面网元，在业务处理过程中，对表项的操作（增删改）非常密集。建议支持巨页内存，以减少内存访问带来的性能损耗。

（2）内存访问性能。内存的访问性能将会直接影响到业务系统的处理性能，因此需要采用必要的机制来保证内存访问性能，有效提升页表查询时命中虚拟寻址缓存的页表；NFVI 应当支持预留连续内存资源池，以减少或消除虚拟机内存碎片；NFVI 需支持将虚拟机的 CPU 和内存部署在同一个 NUMA（non uniform memory access，非一致性内存访问）内，从而降低内存访问的时延。

（3）内核处理实时性。传统的操作系统中，进程都是按照时间片来执行，一个具有更高优先级的用户任务也必须等待前一进程时间片到期后才能被执行。而电信网元要求应用进程的响应时间是确定的，可以通过对进程分配不同的优先级来指定或调整响应时间。因此，NFVI 应当能够支持内核实时性扩展技术，充分利用资源之间的亲和性关系实现计算处理能力的高效发挥，满足电信业务的实时性要求，重点可以考虑虚拟机 vCPU 核绑定、虚拟机 vCPU 超线程、虚拟机 OVS、虚拟机组互斥、虚拟机主机组亲和性等。

2. 可靠性

NFV 的可靠性需要 NFVI 逐层支撑保证，并通过垂直集成增强整系统的可靠性。NFVI 通常通过以下几方面技术保证电信级可靠性。

（1）故障检测。电信业务的故障检测时间一般要求在秒级，要求 NFVI 同样支持秒

级的故障检测，支持在检测硬件故障后实现快速无缝切换受影响的虚拟机业务到状态正常的硬件，减少对电信业务的影响。

（2）故障定界。在 NFV 软硬件解耦的架构下，电信业务软件需要结合 NFVI 基础设施平台信息进行故障定界。NFVI 需要具备定界 COTS 硬件或自身软件故障的能力，同时对受影响的虚拟化资源信息进行上报，以便电信业务软件统一运维管理。

（3）虚拟机智能化部署。NFVI 应当支持智能化的虚拟机部署，满足电信业务无单点故障的需求，并支持电信业务软件的跨站点部署，可跨站点管理不同数据中心的物理资源，满足地理容灾要求。

（4）虚拟机迁移。当物理机出现故障时，NFV 应支持虚拟机可快速迁移和恢复，至少应支持跨主机或跨机架的迁移。

（5）冗余备份。NFVI 应具备硬件和软件的冗余备份能力。硬件方面，支持冗余电源接入、冗余网卡连接、冗余散热风扇等高可用性部件，避免由电源及网络接入等单点故障影响业务功能。交换模块需要支持冗余设计，任何一个部件故障不影响带宽和业务性能。软件方面，应支持虚拟机的快速恢复，对于管理软件需要实现组件级的备份。

（6）硬件高可靠。服务器内存支持高级 ECC（error checking and correcting，错误检查和纠正）内存保护技术或内存镜像保护；磁盘支持 RAID 技术，保证磁盘系统的高可靠性，提高持续工作而不发生故障的能力；网络端口支持聚合或故障切换功能，满足系统对于网络可靠性的需求。

3．转发性能

电信业务存在高吞吐和实时性要求，尤其是针对需要进行编解码转换、协议转换的数据面网元，采用虚拟化技术基于通用硬件和虚拟层软件通常会存在性能瓶颈，无法满足数据面高吞吐量要求。为了在 NFVI 上减少报文调度次数，降低网络转发延迟，提高网络吞吐量，需要提供转发性能加速机制。现有的加速机制包括：

（1）SR-IOV（single root I/O virtualization）：一种基于硬件的虚拟化通道技术。虚拟机直接连接到物理网卡上，获得等同于物理网卡的 I/O 性能和低时延，且多个虚拟机之间高效共享物理网卡。

（2）DPDK（data plane developers kit）：一种内核旁路机制。为了提高包处理的速度，允许虚拟交换机旁路内核并直接与兼容的网卡通信。

（3）超线程技术。要求 COTS 硬件支持超线程技术提高 CPU 的并发处理数，使得单个处理器可使用线程级并行计算，减少 CPU 的闲置时间。

（4）硬件加速机制。NFV 技术采用通用硬件来承载电信业务，对于某些特殊业务采用上述（1）～（3）的软件加速方式而不是专有硬件处理，会导致转发性能的显著下降，因此需要引入硬件加速技术来解决相关业务的转发性能问题。目前业界主流的专用硬件加速技术包括通用加速资源池、专用 PCI 加速卡或 CPU 内置加速芯片等，但具体采用何种专用硬件加速技术仍有待进一步研究评估。

4．组网

早期数据中心主要满足外部对数据中心的访问，因此流量以南北向为主。电信业务

多为分布式部署，网元间的流量较多，随着NFV的部署，电信业务对数据中心的流量模型将产生巨大的冲击，数据中心流量将由南北向为主转变为东西向为主。组网上需要考虑如何保障电信业务的接入质量。

对于NFV而言，NFVI需要加强接入质量的保障，确保用户在访问数据中心时带宽能够得到保证，因此首先要求COTS硬件提供足够数量的40GE/100GE以太网接口，并支持JumboFrame提升报文传输的效率，从而保障电信业务的南北向流量的QoS。电信业务的东西向流量在数据中心的VNF之间交互转发，将导致数据中心内的东西向流量相比南北向流量成倍增长。NFV的内部组网同样要求大带宽的QoS保障，而数据中心的虚拟机之间一般采用二层组网以支持动态迁移，因此这将对承载数据中心内东西向流交换的交换机性能提出要求，其需要支持高达40GE～400GE每服务器节点的交换组网能力。

5. 安全

NFVI基础设施平台资源的安全主要涉及电信业务运行的物理硬件资源、虚拟化平台和网络的安全可靠。对于物理硬件资源，需要保证COTS硬件的安全可信。COTS硬件需要提供基于硬件芯片的可信环境，以便于虚拟层及业务软件层可基于硬件可信环境构筑信任根，实现安全启动和安全存储。目前通过使用可信赖的平台模块(trusted platform module，TPM)来完成加密、签名、认证、密钥生成等功能。对于接口的访问，COTS硬件需要进行访问控制和认证鉴权，防止非法访问。对于虚拟化平台资源，要求提供租户、虚拟机以及不同业务的安全隔离，避免虚拟机之间的数据窃取或恶意攻击。对于虚拟化平台及其所管理的虚拟资源的访问，要求提供访问控制和认证鉴权，防止非法访问对系统的影响。在存储方面，对物理存储实体的直接访问具备禁止或限制的能力，提供虚拟存储数据清除、虚拟存储数据审计、虚拟存储数据访问控制和冗余备份功能；在计算方面，能利用加密算法协处理器，高效实现VNF内部的机密性和完整性保护。对于网络资源，应具备对虚拟机的隔离和访问控制能力，虚拟机之间需要授权和鉴权后才能建立通信连接。通过在线的深度报文检测、虚拟防火墙技术以及基于Netflow采集的流量溯源与关联分析等综合技术，可以建立基本的NFV网络安全防护体系，能够支持抵御畸形报文攻击、DoS攻击和仿冒攻击等。

6. 环境适应性

国际标准组织对电信设备的环境适应性有正式的规定，例如，欧洲的EN3000019和EN300318系列要求、北美的NEBS系列要求等。电信业务的设备要求普遍高于目前COTS设备的工作指标。为了使COTS设备系统稳定可靠地运行，减少机器故障对电信业务的影响，COTS设备需满足一系列的环境因素要求，包括工作的温度、湿度、散热方式、高度、供电模式等。

7. 能耗

NFVI中通用服务器功耗相较于传统电信设备功耗更大，需要提供相应的手段和措施尽量降低能耗，减少NFVI投资和运维管理费用。通常可以采用以下几种技术和措施。

（1）功耗测量。支持连续测量系统功耗，并应能够根据系统负载控制 CPU 功耗状态，且不会造成性能损失。支持带外功耗实时监控能力，支撑主动策略型节能管理。

（2）电源封顶技术。可以通过服务器的动态配置或功率封顶，有效地对每一台服务器能耗进行准确控制。

（3）功耗控制。支持平台功耗控制，通过 IPMI 标准协议，将单台、多台服务系统或整个弹性计算平台功耗设置为限定目标功率，并能在该功耗限定下达到最佳性能。

（4）功耗报警。支持功耗阈值警报，通过动态监控限定目标的功耗来监控平台的功耗，当无法维持限定目标功耗值时，系统能向管理平台发送警报。

7.9.3 传统电信网络向 NFV 的演进策略

传统电信运营商正在加速战略转型，NFV 成为转型的重要技术支撑。随着电信业务和互联网业务的相互渗透与融合，电信运营商开始迈入信息服务转型的关键发展阶段，由此带来网络基础架构的深刻变革，构建一张“资源可全局调度、能力可全面开放、容量可弹性伸缩、架构可灵活调整”的新一代网络成为运营商网络演进发展的重要方向。

在 NFV 领域，国内运营商积极参与国际标准开源组织的标准化工作，相继成立开放实验室开展 NFV 相关专业的功能、性能和互通性测试验证工作，促进 NFV 生态系统构建，推动产业发展成熟。

在 NFV 商用化部署方面，将遵循以业务为驱动，从行业应用到公众应用，先局部后整体，先控制面后用户面的逐步演进、迭代升级的思路，新建网络在满足性能的前提下优先考虑以 NFV 形式部署。NFV 商用目标方案是硬件、虚拟资源层、上层网元功能三层全部解耦，但在商用初期，鉴于虚拟资源层和上层网元功能解耦难度较大，可采用软硬件两层解耦的模式作为过渡。

NFV 的部署将是一个渐进的过程，传统网络和 NFV 长期共存。在全网部署 NFV 前，运营商现网设备不可能全部立刻退网，因此 NFV 的部署将是一个渐进的长期过程。NFV 的部署范围将覆盖核心网、承载网、接入网等全领域。其典型部署场景包括：

（1）基于 NFV 提供新业务，如基于蜂窝的窄带物联网 NB-IOT（narrow band internet of things）、Gi-LAN 等。通过部署 VNF 来提供新业务具有部署周期短、对现网影响小、业务创新快的优势。

（2）行业或企业应用，如公共安全（public safety）、BYOD 等。这些应用对网络部署和功能有特殊要求，如公共安全一般要求资源隔离、安全增强和高优先级服务等，可以利用 NFV 的多实例/切片技术方便地为这些应用提供服务。

（3）网络扩容，如通过 vEPC 扩容。面对快速的业务增长，运营商通过部署 VNF 来分担现网 PNF（physical network function，物理网络功能）的负载，同时逐步淘汰现网设备，实现网络的平滑演进。

（4）网络升级替换。传统设备因为折旧或需要进行硬件替换时，通过引入 NFV 设备，可以加快未来网络的升级部署速度。

目前 NFV 还未完全成熟商用，各虚拟网络功能 VNF 的复杂度和技术要求也不相同，所以部署中要考虑各 VNF 的成熟度。偏重计算和存储类的 VNF（如 CSCF、AS、

PCRF、HSS 等)相对成熟,偏重转发和媒体处理的 VNF(如 S/P-GW、SBC)成熟度低。所以通常的部署节奏是先业务(如 AS、RCS)和控制面网元(如 IMS、MME),再转发和媒体层(如 EPC-GW、SBC)。

从业务和设备的生命周期角度考虑,一般新业务或成长期业务(如 Gi-LAN)涉及的功能优先虚拟化,衰退期业务(如 CS)涉及的功能虚拟化要慎重;存量设备也存在替换周期,一般即将退网设备优先考虑虚拟化。总之,NFV 的部署一般原则是"需求导向,业务驱动,效率优先,从易到难"循序渐进的过程。下面是传统网络与 NFV 协调部署策略[18]。

1. PNF 和 VNF 共存部署场景

虚拟网络功能(VNF)和物理网络功能(PNF)共存部署的总体目标是:充分利用现网投资,最小化网络风险,实现业务平滑迁移。目前存在两种典型的共存部署场景。

1)在传统网络基础上采用 NFV 开展新业务

在现网基础上新建设 VNF,并由 VNF 提供新业务或服务特定用户群(如 MVNO 用户),如图 7-13 所示。这种方式对现有网络不产生影响,资源和管理面独立,但业务互通,具有部署快、风险小的特点,比较适合由 NFV 提供新业务或小规模商用场景。图 7-13 所示是在现有传统 EPC 网络基础上开展 Gi-LAN 的示例。运营商有两种选择,采用传统物理设备部署 Gi-LAN 业务,或者采用 NFV 部署 Gi-LAN 业务。

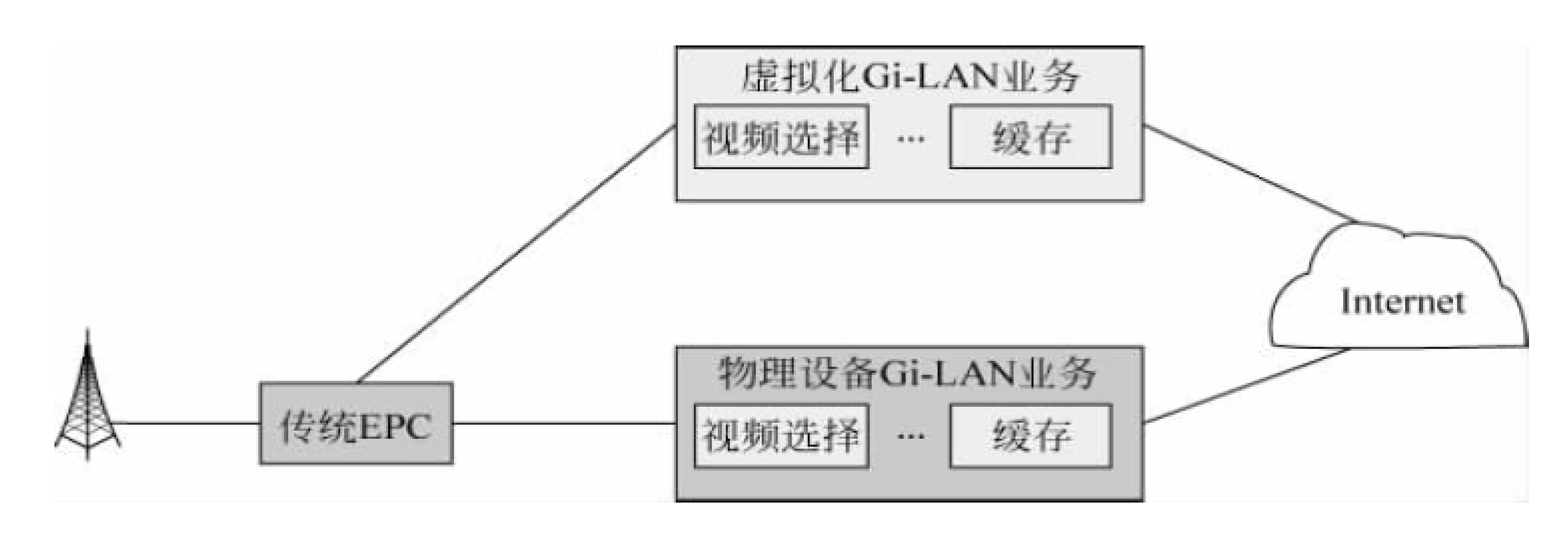


图 7-13 传统网络部署 NFV 新业务

2)对传统网络采用 NFV 扩容现有业务

现网 PNF 池内增加 VNF 网元。如图 7-14 所示是 vEPC 和现网 EPC 混合池部署示例。这种组网具有如下的优势:

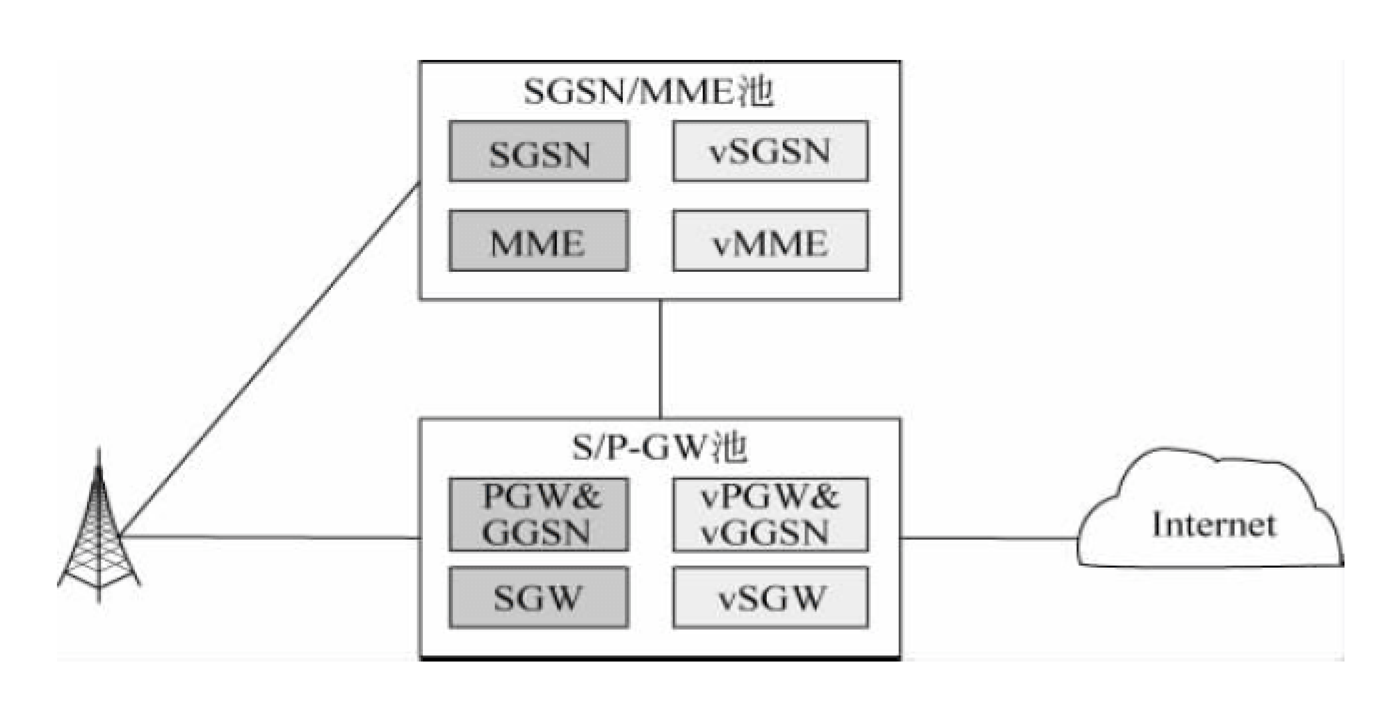


图 7-14 采用 NFV 扩容现有业务

(1) 异构平台组池,互为备份,可靠性更高,可以避免全网故障的风险。

(2) 用户和业务平滑迁移至 VNF,迁移过程可以随时终止和回退,最小化网络割接风险。

(3) 充分利用旧现网 PNF,保护现有投资,并可逐步淘汰 PNF,实现网络平滑演进。

(4) 利用 VNF 实现池的弹性伸缩,当池内业务量增加时,VNF 自动扩容吸收话通话业务,应对业务突发和信令风暴。

2. VNF 和 PNF 业务特性一致

为了使 VNF 能平滑承接现网业务,VNF 和现网 PNF 的特性对齐非常重要。原则上 VNF 应满足如下需求:

(1) VNF 的部署对终端和用户业务是完全透明的,即网络功能虚拟化不应该影响用户业务和体验。

(2) VNF 应该支持所有现网接口,并且可以与 PNF 互操作。

(3) 电信业务规范本身是平台无关的,理论上,只要 VNF 遵循相关规范并继承 PNF 的业务特性就可以承接现网业务,但实际部署中需要解决如下问题。

① VNF 转发性能优化。虚拟化后,进出 VNF 报文的需经过虚拟层(如 vSwitch)复制和转发,影响转发性能,特别是时延/抖动,可能会影响用户的业务体验,需要优化方案。

② VNF 基于通用 COTS,采用全 IP 接口,现网非 IP 接口(如 E1/ATM)需要提前考虑 IP 化或部署 IWF(Interworking function,网络互联功能)进行转换。

③ VNF 可以灵活部署和迁移,业务流量不固定于某物理端口/线路。现网探针一般采用链路分光/端口镜像方式实现引流,该方案不再适用虚拟化场景,需重新考虑部署和组网方案。

3. VNF 和 PNF 网元协同管理

PNF 软件和硬件集成在一个设备中,设备生命周期、业务配合、日常维护等均通过 EMS 完成。NFV 要求业务自动部署,软硬件分层运维。新增 MANO 负责网络业务(NS)和 VNF 的生命周期管理以及全局资源视图的管理。VNF 的业务配置、业务策略管理、日常维护等仍然由 EMS 负责,MANO 和 EMS/OSS 通过运营商自定义的协同方式完成对 VNF 的全面管理。在 PNF 和 VNF 混合组网情况下,网络侧应支持对 PNF 和 VNF 网元的协同管理,以实现端到端网络服务的管理。

4. 垂直运维和分层运维模式共存

NFV 的出现,尤其是 PNF、VNF 混合网共存环境,对网络运维提出了较大挑战。传统网络软硬件一体化,按业务构建烟囱式运维团队;NFV 后软硬件解耦,网络运维团队面临转型,可能需要分层构建运维团队,同时需跨层协同运维,变化主要体现在以下两方面,一是各实体功能网元演变为以软件形态存在的虚拟网元,传统分业务领域的维护依然存在,而且网络业务和网元还可能分层运维,但是不再针对设备硬件进行维护;二是新增加 NFV 基础设施维护,管理和维护各数据中心的硬件和虚拟资源。另外,在技术上也

面临如下挑战：

（1）混合组网场景下，业务涉及 PNF 和 VNF，这就要求跨 PNF/VNF、跨域、跨厂商的端到端业务编排和保障。

（2）未来大流量将聚集在视频、重大事件/突发事件等场景，流量时空不均衡、潮汐效应显著，需研究如何利用 NFV，实现存量资源和虚拟化资源统一管理和调度，避免按峰值规划。

参考文献

[1] 杨宇. 网络虚拟化资源管理及虚拟网络应用研究[D]. 北京：北京邮电大学，2013.

[2] 温涛，虞红芳，李乐民. 专题：网络以及功能虚拟化——网络虚拟化的过去、现在和未来[J]. 中兴通讯技术，2014(3)：1-7.

[3] 冯志勇，冯泽冰，张奇勋. 无线网络虚拟化架构与关键技术[J]. 中兴通讯技术，2014(3)：16-21.

[4] 曹进军. 基于网络切片的信息聚合研究[J]. 情报理论与实践，2013，36(9)：104-107.

[5] 韩言妮，覃毅芳，慈松. 未来网络虚拟化关键技术研究[J]. 中兴通讯技术，2011，17(2)：15-19.

[6] 刘文志. 网络虚拟化环境下资源管理关键技术研究[D]. 北京：北京邮电大学，2012.

[7] 卢波. 虚拟网络映射策略与算法研究[D]. 北京：北京邮电大学，2014.

[8] 刘川川. 无线网络虚拟化中资源分配算法研究[D]. 湖南：湖南大学，2013.

[9] 宁尚明. 云计算环境中虚拟化技术的研究探讨[J]. 电子制作，2015(85).

[10] Network Functions Virtualization—Introductory White Paper. Definition. http://portal.etsi.org/NFV/NFV_White_Paper.pdf.

[11] Network Functions Virtualization—Update White Paper. NFV Architectural Framework Document. http://portal.etsi.org/NFV/NFV_White_Paper2.pdf.

[12] Mijumbi, Rashid, et al. Management and orchestration challenges in network functions virtualization. IEEE Communications Magazine 2016，54(1)：98-105.

[13] Network Functions Virtualization White Paper #3. NFV Reliability and Availability. http://portal.etsi.org/NFV/NFV_White_Paper3.pdf.

[14] Network Functions Virtualization White Paper #3. Fields of Application and Use Case. http://portal.etsi.org/NFV/NFV_White_Paper3.pdf.

[15] Gonzalez, Andres, et al. Service Availability in the NFV Virtualized Evolved Packet Core. 2015 IEEE Global Communications Conference (GLOBECOM). IEEE，2015.

[16] Bronstein, Zvika, and Eyal Shraga. NFV virtualization of the home environment. 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC). IEEE，2014.

[17] Chi, Po-Wen, Yu-Cheng Huang, and Chin-Laung Lei. Efficient NFV deployment in data center networks. 2015 IEEE International Conference on Communications (ICC). IEEE，2015.

[18] NFV 产业发展白皮书 2016. SDN/NFV 产业联盟.

第8章 异构无线网络仿真平台

本章总结了异构无线网络仿真平台的相关技术与实现，简要介绍了 NS-2 网络模拟器及仿真分析所需要的相关工具，对支持 UMTS/WLAN 双模的网络仿真平台及相关仿真实例的实现进行了详细阐述。

8.1 NS-2 网络模拟器

为了验证网络协议的正确性和进行相关性能测试，人们提出了很多方法，而目前最广泛使用的就是通过虚拟环境进行模拟仿真。NS-2 (network simulator，version 2)是最流行的进行网络模拟的软件之一。它是一种面向对象的网络仿真器，其本质是一个离散事件模拟器——它本身有一个虚拟时钟，所有的仿真都由离散事件驱动。

8.1.1 NS-2 简介

目前 NS-2 可以用于仿真各种不同的 IP 网，已经实现的一些仿真有：网络传输协议，如 TCP 和 UDP；业务源流量产生器，如 FTP、Telnet、Web CBR 和 VBR；路由队列管理机制，如 Droptail、RED 和 CBQ；路由算法，如 Dijkstra 等。NS-2 也为进行局域网的仿真而实现了多播以及一些 MAC 子层协议。

NS-2 最早来源于 1989 年的 Real Network Simulator 项目，经过多年的发展后，于 1995 年得到施乐公司(Xerox)的支持，加入 VINT 项目。NS-2 一直以来都在吸收全世界各地研究者的成果，包括 UCB、CMU 等大学和 Sun 等公司的无线网络方面的代码。

8.1.2 NS-2 仿真原理

NS-2 通过对离散事件的模拟来对网络进行仿真，在虚拟时钟的作用下，各个模块事件的相互作用模拟了网络的行为。

1. NS-2 使用的语言

NS-2 使用 C++ 和 OTCL 作为开发语言。之所以选择这两种语言，是因为模拟器有两方面的事情需要做：一方面，具体协议的模拟和实现，需要一种程序设计语言，能够高效率地处理字节、报头等信息，能够应用合适的算法在大量的数据集合上进行操作。为了实现该任务，程序内部模块的运行速度是非常重要的，而运行模拟环境的时间、寻找和修复 Bug 的时间，以及重新编译和运行的时间不是很重要。在这种情况下，C++ 语言是非常合适的。另一方面，许多网络中的研究工作都是围绕网络组件和环境的具体参数的设置和改变而进行的，需要在短时间内快速地开发和模拟出所需要的网络环境，并且方便修改和发现、修复程序中的 Bug。在这种要求下，网络环境布置的时间就显得很重要了，因为模拟环境的建立和参数的配置只需要运行一次。因此，脚本语言就具有很大优势，具有面向对象特性的 TCL 语言脚本可以充分满足此需求。

NS-2 可以说是 OTCL 的脚本解释器，它包含仿真事件调度器、网络组件对象库以及网络构建模型库等。事件调度器计算仿真时间，并且激活事件队列中的当前事件，执行一些相关的事件，网络组件通过传递分组来相互通信，但这并不耗费仿真时间。所有需要花费仿真时间来处理分组的网络组件都必须要使用事件调度器。它先为这个分组发出一个事件，然后等待这个事件被调度回来之后，才能做下一步的处理工作。事件调度器的另一个用处就是计时。NS-2 是用 OTCL 和 C++ 编写的。由于效率的原因，NS-2 将数据通道和控制通道的实现相分离。为了减少分组和事件的处理时间，事件调度器和数据通道上的基本网络组件对象都使用 C++ 写出并编译，这些对象通过映射对 OTCL 解释器可见。

当仿真完成以后，NS-2 将会产生一个或多个基于文本的跟踪文件。只要在 TCL 脚本中加入一些简单的语句，这些文件中就会包含详细的跟踪信息。这些数据可以用于下一步的分析处理，也可以使用 NAM 将整个仿真过程展示出来。

2. 使用 NS-2 进行网络仿真的方法和一般过程

进行网络仿真前，首先分析仿真涉及哪个层次，NS-2 仿真分两个层次：一个是基于 OTCL 编程的层次。利用 NS-2 已有的网络元素实现仿真，无须修改 NS-2 本身，只需编写 OTCL 脚本。另一个是基于 C++ 和 OTCL 编程的层次。如果 NS-2 中没有所需的网络元素，则需要对 NS-2 进行扩展，添加所需网络元素，即添加新的 C++ 和 OTCL 类，编写新的 OTCL 脚本。

假设用户已经完成了对 NS-2 的扩展，或者 NS-2 所包含的构件已经满足了要求，那么进行一次仿真的步骤大致如下：

(1) 开始编写 OTCL 脚本。首先配置模拟网络拓扑结构，此时可以确定链路的基本特性，如延迟、带宽和丢失策略等。

(2) 建立协议代理，包括端设备的协议绑定和通信业务量模型的建立。

(3) 配置业务量模型的参数，从而确定网络上的业务量分布。

(4) 设置 Trace 对象。NS-2 通过 Trace 文件来保存整个模拟过程。仿真完后，用户可以对 Trace 文件进行分析研究。

（5）编写其他的辅助过程，设定模拟结束时间，至此 OTCL 脚本编写完成。

（6）用 NS-2 解释执行刚才编写的 OTCL 脚本。

（7）对 Trace 文件进行分析，得出有用的数据。

（8）调整配置拓扑结构和业务量模型，重新进行上述模拟过程。

NS-2 采用两级体系结构，为了提高代码的执行效率，NS-2 将数据操作与控制部分的实现相分离，事件调度器和大部分基本的网络组件对象后台使用 C++ 实现和编译，称为编译层，主要功能是实现对数据包的处理；NS-2 的前端是一个 OTCL 解释器，称为解释层，主要功能是对模拟环境的配置、建立。从用户角度看，NS-2 是一个具有仿真事件驱动、网络构件对象库和网络配置模块库的 OTCL 脚本解释器。NS-2 中编译类对象通过 OTCL 连接建立了与之对应的解释类对象，这样用户间能够方便地对 C++ 对象的函数进行修改与配置，充分体现了仿真器的一致性和灵活性。

3. NS-2 的功能模块

NS-2 仿真器封装了许多功能模块，最基本的是节点、链路、代理、数据包格式等。下面分别来介绍一下各个模块。

（1）事件调度器。目前 NS-2 提供了 4 种具有不同数据结构的调度器，每种都是由不同的数据结构实现的，分别是链表、堆、日历表和实时调度器。调度器通过选择下一个最早时间，执行完它，以及返回继续执行下一个事件的方式工作。其主要功能是：处理分组的时延以及充当定时器。

（2）节点（node）。节点是由 TclObject 对象组成的复合组件，在 NS-2 中可以表示端节点和路由器，不是 TCLObject 类的派生类，而是 NS-2 中独立的类，一个节点本质上是一个分类器的集合。

（3）链路（link）。链路是网络拓扑的第二部分，是 NS-2 复合网络组件的另一个大类。由多个组件复合而成，用来连接网络节点。所有的链路都是以队列的形式来管理分组的到达、离开和丢弃。

（4）代理（agent）。代理代表了网络层的分组的起点和终点，并被用于各层协议的实现。负责网络层分组的产生和接收，也可以用在各个层次的协议实现中。Agent 类是由 C++ 和 OTCL 共同实现的。每个 agent 连接到一个网络节点上，由该节点给它分配一个端口号。

（5）包（packet）。包由头部和数据两部分组成。一般情况下，packet 只有头部，没有数据部分。

8.1.3 NS-2 的扩展：添加新协议

本节将用实例的形式展示如何使用 NS-2 来实现一个新协议。

实现一个简单的 ping 协议：发送节点可以发送数据包给接收节点，另一个节点——接收节点直接将该数据包返回，由此可以得到一个往返时延（round-trip-time，RTT）。

1. 头文件

新的 ping 数据包文件头应该包含该数据包携带数据的相关信息。那么，首先需要在

ping.h 头文件中对这个文件头进行声明。

```
struct hdr_ping {
  char ret;
  double send_time;
};
```

当该数据包是由发送节点发往接收节点时，字符 ret 将被设置为 0，当数据包从接收节点返回发送节点时，字符 ret 将被重新设置为 1。send_time 是数据包发送的时间戳，该变量用于计算往返时延。

声明一个 Agent 类的子类 PingAgent：

```
class PingAgent : public Agent {
  public:
    PingAgent();
    int command(int argc,const char * const * argv);
    void recv(Packet * ,Handler * );
  protected:
    int off_ping_;
};
```

接下来，将使用 C++代码实现这个类和类中函数（类中数值变量 off_ping_用于访问数据的头文件）。

2. C++代码

首先必须定义 C++代码与 TCL 代码之间的链接：

```
static class PingHeaderClass : public PacketHeaderClass {
public:
  PingHeaderClass() : PacketHeaderClass("PacketHeader/Ping",
                                        sizeof(hdr_ping)) {}
static class PingHeaderClass : public PacketHeaderClass {
public:
  PingHeaderClass() : PacketHeaderClass("PacketHeader/Ping",
                                        sizeof(hdr_ping)) {}
} class_pinghdr;

static class PingClass : public TclClass {
public:
  PingClass() : TclClass("Agent/Ping") {}
  TclObject * create(int,const char * const * ) {
    return (new PingAgent());
  }
} class_ping;
```

接下来，需要实现类 PingAgent 的构造器。该构造器必须绑定 C++与 TCL 共同访问的变量：

```
PingAgent::PingAgent() : Agent(PT_PING)
```

```
{
  bind("packetSize_",&size_);
  bind("off_ping_",&off_ping_);
}
```

当类 PingAgent 的 TCL 命令被执行时，command()函数将会被执行。由于协议的任务是将一个 ping 数据包从一个 Agent 发往另一个 Agent，所以在实例中该 TCL 命令将会是'$ pa send'(pa 为一个 Agent/Ping 类的实例)。需要在 command()函数中描述该命令，如果在 command()函数中找不到匹配的命令，那么该命令将会传递到父类中去寻找匹配的命令。函数实现如下：

```
int PingAgent::command(int argc,const char * const *  argv)
{
  if (argc == 2) {
    if (strcmp(argv[1],"send") == 0) {
      //建立数据包
      Packet * pkt = allocpkt();
      //数据包头文件访问地址
      hdr_ping * hdr = (hdr_ping * )pkt->access(off_ping_);
      //将 off_ping_置零,告知接收端该数据包为发送数据包
      hdr->ret = 0;
      //存储时间戳
      hdr->send_time = Scheduler::instance().clock();
      //发送数据包
      send(pkt,0);
      //返回 TCL_OK,来告知回调函数该命令已经执行完毕
      return (TCL_OK);
    }
  }
  //如果该命令不能由 PingAgent()::command 处理,调用父类中的 command()函数
  return (Agent::command(argc,argv));
}
```

函数 recv()定义了类在接收到数据包时的行为。当 ret 为 0 时，数据包的时间戳不变，将 ret 置 1，同时将数据包返回发送节点。当 ret 为 1 时，程序将会调用 TCL 函数来处理该事件。函数实现如下：

```
void PingAgent::recv(Packet * pkt,Handler * )
{
  //访问数据包的 IP 头
  hdr_ip * hdrip = (hdr_ip * )pkt->access(off_ip_);
  //访问数据包的 ping 头
  hdr_ping * hdr = (hdr_ping * )pkt->access(off_ping_);
  //判断 ret 是否为 0
  if (hdr->ret == 0) {
    //存储旧时间戳
    double stime = hdr->send_time;
    //销毁数据包
    Packet::free(pkt);
    //新建新数据包
```

```
        Packet * pktret = allocpkt();
        //访问新数据包的 ping 头
        hdr_ping * hdrret = (hdr_ping * )pktret->access(off_ping_);
        //将 ret 置 1
        hdrret->ret = 1;
        //设置时间戳
        hdrret->send_time = stime;
        //Send the packet
        send(pktret,0);
    } else {
        //调用 TCL 函数来解释结果
        char out[100];
        //计算往返时延
        sprintf(out,"%s recv %d %3.1f",name(),
                hdrip->src_.addr_>> Address::instance().NodeShift_[1],
                (Scheduler::instance().clock() - hdr->send_time) * 1000);
        Tcl& tcl = Tcl::instance();
        tcl.eval(out);
        //销毁数据包
        Packet::free(pkt);
    }
}
```

以上代码已经完成了协议中 C++的类实现,接下来需要在 NS 的源文件中来设置一些参数。

3. 必要参数设置

如果要增加代理,特别是要使用行的数据包格式时,必须改变 NS 设置。首先,编辑 packet.h,在该文件下可以找到数据包协议的声明(如 PT_TCP、PT_TELNET 等)。在该文件中添加一个新的协议 PT_PING:

```
enum packet_t {
    PT_TCP,
    PT_UDP,
        ...
    //insert new packet types here
    PT_TFRC,
    PT_TFRC_ACK,
        PT_PING,                    //新协议的 ID
    PT_NTYPE                        //该 ID 必须位于末尾
};
```

编辑同文件下的 p_info()将协议包含进来:

```
class p_info {
public:
    p_info() {
        name_[PT_TCP] = "tcp";
        name_[PT_UDP] = "udp";
                ...
```

```
            name_[PT_TFRC] = "tcpFriend";
            name_[PT_TFRC_ACK] = "tcpFriendCtl";
                    name_[PT_PING] = "Ping";
            name_[PT_NTYPE] = "undefined";
        }
            ...
}
```

在编译之前，应该执行一次 make depend 来保证两个文件的重新编译。

接下来，编辑 tcl/lib/ns-default.tcl 文件。该文件定义了 TCL 所有默认参数。插入如下行来定义数据包大小：

```
Agent/Ping set packetSize_64
```

在该文件开头的列表中，插入新数据包的入口：

```
{ SRMEXT off_srm_ext_}
        { Ping off_ping_}} {
set cl PacketHeader/[lindex $ pair 0]
```

最后，需要更改 Makefile，将 ping.o 文件加入 ns 的对象列表中。插入后的文件末尾如下：

```
sessionhelper.o delaymodel.o srm - ssm.o \
srm - topo.o \
ping.o \
$ (LIB_DIR)int.Vec.o $ (LIB_DIR)int.RVec.o \
$ (LIB_DIR)dmalloc_support.o \
```

4. TCL 代码

下面将介绍 TCL 中 recv 处理过程的实现。当收到一个返回的数据包时，C++代码中的 recv()函数会调用该过程。

```
Agent/Ping instproc recv {from rtt} {
        $ self instvar node_
        puts "node [ $ node_id] received ping answer from \
              $ from with round - trip - time $ rtt ms."
}
```

通过以上工作，实现了一个简单的 ping 协议。实际工作中，即便是复杂的协议实现，其添加过程都遵循着类似的流程和框架。

8.2 仿真分析需要的相关工具

8.2.1 数据处理工具 grep 和 gawk

用 NS 进行仿真时，首先新建一个 TCL 文件，在文件中搭建仿真环境，定义各个网络节点，规定各层使用的协议，定义各种业务流和模拟环境的各类参数等。然后在命令行

下使用 NS 命令运行模拟，得到 trace 文件。模拟过程中得到的仿真数据被记录在 trace 文件中。每次运行 NS 生成的 trace 结果输出文件可能非常巨大，依靠人去一行行读取几乎是不可能的，如何分析这么大的数据量往往是一个难题。特别是在协议的改进或调试过程，通过结果分析很快找出问题所在，才能继续完善设计和排除 Bug。这时就需要用一些小工具来对结果进行分析。例如，对下面的 NS 运行结果 result.tr 进行分析，trace 文件结构类似于：

```
s 10.000000000 _0_AGT --- 0 cbr 512 [0 0 0 0] ------- [0:0 2:0 32 0] [0] 0 0
r 10.000000000 _0_RTR --- 0 cbr 512 [0 0 0 0] ------- [0:0 2:0 32 0] [0] 0 0
s 10.000000000 _0_RTR --- 0 cbr 532 [0 0 0 0] ------- [0:0 2:0 30 0] [0] 0 0
r 10.004772500 _1_RTR --- 0 cbr 532 [0 ffffffff 0 800] ------- [0:0 2:0 30 0] [0] 1 0
f 10.005056593 _1_RTR --- 0 cbr 532 [0 ffffffff 0 800] ------- [0:0 2:0 29 0] [0] 1 0
r 10.009829093 _2_AGT --- 0 cbr 532 [0 ffffffff 1 800] ------- [0:0 2:0 29 0] [0] 2 0
r 10.009829093 _0_RTR --- 0 cbr 532 [0 ffffffff 1 800] ------- [0:0 2:0 29 0] [0] 2 0
D 10.009829093 _0_RTR LOOP 0 cbr 532 [0 ffffffff 1 800] ------- [0:0 2:0 29 0] [0] 2 0
```

首先想判断一下节点 0 发出的包，经过节点 1 转发后，是否全被节点 2 接收。这只需要统计一下节点 0 发送的数据量和节点 2 接收的数据量。很简单，可以采用下面的 Linux 命令来统计：

把应用层接收到的数据的输出行都写到 g_r 文件中：

```
grep "r.*AGT" result.tr > g_r
```

把应用层发送的数据的输出行都写入 g_s 文件中：

```
grep "s.*AGT" result.tr > g_s
```

统计文件里的字符个数、单词个数、行数：

```
wc g_?
```

输出结果为：

```
50   1000   4507   g_r
50   1000   4507   g_s
100  2000   8564   total
```

从输出的结果可以看出，0 节点发送了 50 个包(g_r 文件共 50 行)，2 节点接收了 50 个包，所以 2 节点接收了 0 节点发出的所有数据包。

实际上，当 trace 输出文件很大时，采用 grep 命令来收集需要的信息时间会较长，而且也不够灵活。常常编写并运行 gawk 脚本，其功能强大而且处理速度快。下面学习 gawk 脚本的编写和运行。

gawk 是一种程序语言，对于资料的处理具有很强的功能，可以使用很短的代码轻易地完成对文本档案的修改、分析、提取和比较等处理。gawk 的主要功能是针对档案的每一行(line)搜寻指定的 patterns。当一行里有符合指定的 patterns，gawk 就会在此行执行被指定的 actions。gawk 依此方式处理输入档案的每一行直到输入档案结束。在 NS 的模拟结果分析中，经常用到 gawk 来进行数据分析和统计等。命令如下：

```
gawk 'program' input-file1 input-file2 …                //程序代码较短时
gawk -f program-file input-file1 input-file2…           //程序代码较长
```

其中，当程序代码较长时，gawk 命令存在一个文件中，即 patterns 与 actions 写在上述名为 program-file 的文件里面。

gawk 程序是由很多的 pattern 和 action 所组成的：pattern {action}。功能是针对文件的每一行搜寻指定的模式(pattern)，当一行里有符合指定的模式时，gawk 就会在此行执行指定的动作(action)。

常用的系统变量包括：

(1) RS——记录分隔符(默认为\n)。

(2) FNR——目前的输入档案已经被读取的记录个数。

(3) NR——所有输入档案已经被读取的记录个数。

(4) FS——栏分隔符(默认为空白字符)。

(5) NF——目前记录有多少个栏位。

(6) OFS——输出栏位分隔符(初始值为空格)。

(7) ORS——输出记录分隔符(初始值为\n)。

常用的 pattern 包括：

(1) /regular expression/——正则表达式，如 exp~/regexp/、exp! ~/regexp/。

(2) /expression/——单一表达式，值不为 0 或字串不空时视为匹配。

(3) pat1，pat2——指定记录的范围。

(4) BEGIN、END——gawk 在开始执行或要结束时会分别执行由 BEGIN 或 END 所指定的 action。

(5) null——对于每个输入记录皆视为符合 pattern。

常用的 action 包括：

(1) expression——算术运算、比较运算、布尔运算、条件运算等。

(2) 控制语句——if、while、do-while、for、break、continue、next、exit。

(3) 内建函数——数值方面的函数(sqrt、exp、log、sin、rand、srand)。

(4) 字串方面的函数(index、length、match、sprintf、sub、gsub、substr)。

(5) 输入/输出的内建函数(close、system)。

(6) 使用者定义的函数。

8.2.2 图形绘制工具

Gnuplot 是一款很适合科技工作者的绘图软件。下面将介绍如何使用 Gnuplot 来帮助我们完成数据可视化的工作。

Gnuplot 对目前主流的 Linux 系统和 Windows 系统都有对应的版本。主流的 Linux 发行版本都可以通过对应的软件工具箱来安装 Gnuplot。在 Windows 下，需要到官方网站下载最新的版本，解压后就可以直接使用。

Gnuplot 通过命令行来完成与用户的交互功能。当启动一个 Gnuplot 终端时，可以看到 Gnuplot 的版本信息。其中，只需要在 gnuplot >提示符后输入相应的命令就可以实现对应的功能。如果需要退出 Gnuplot，只需要输入 quit 命令或者 exit 命令。

1. 绘制简单的图像

Gnuplot 支持几种简单的运算符：+、-、*、/等。使用 x ** y 表示 x 的 y 次方。在提示符右侧输入命令：

```
plot sin(x)
```

就可以绘制出如图 8-1 所示的一个最简单的正弦曲线图。

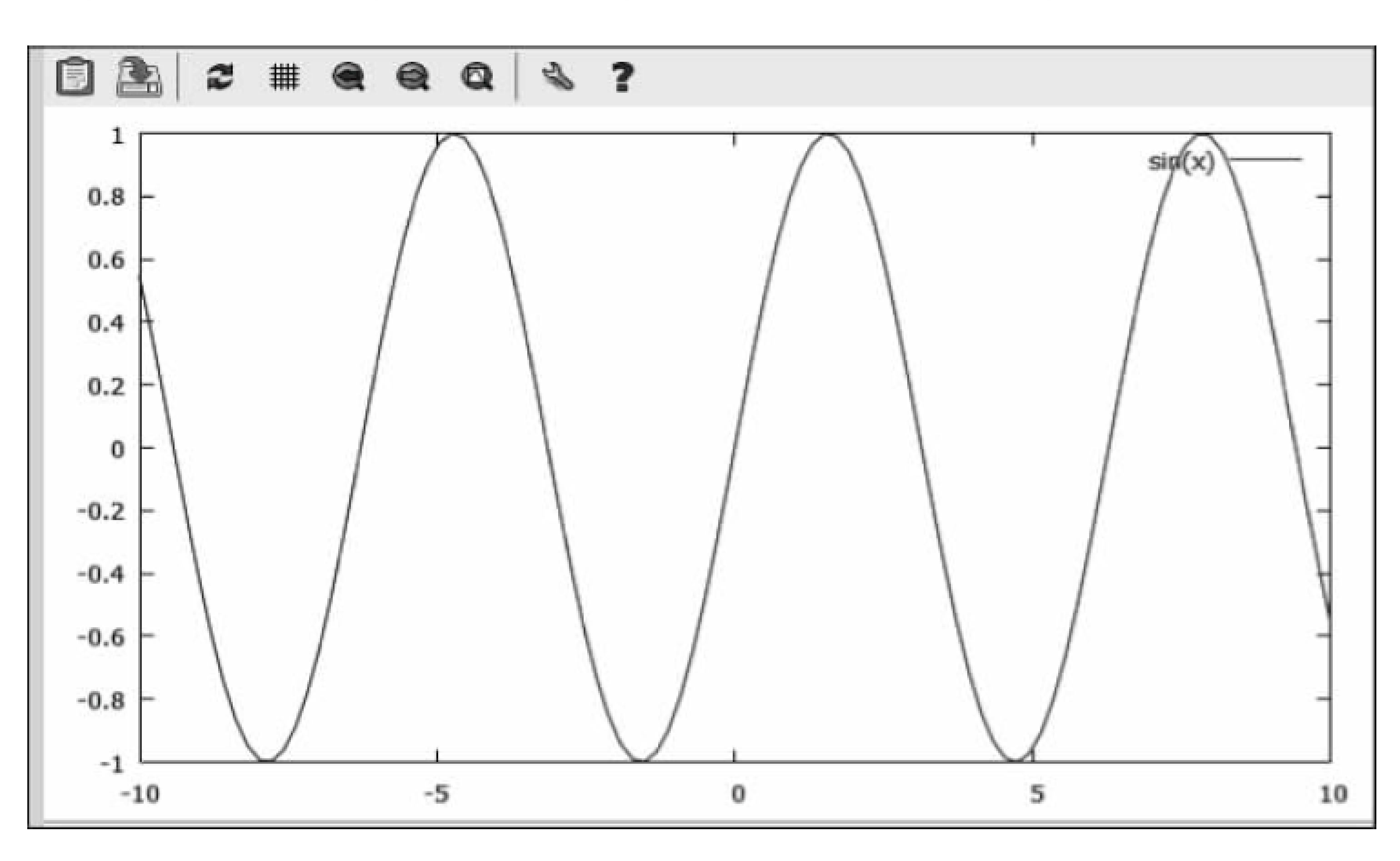


图 8-1 简单正弦曲线

可以使用下面的命令来设置图像的参数：

```
set xrange [ - 10,10]          //设置坐标轴范围为 - 10,10
set title "示例"               //设置图像的标题
set xlabel "x 轴"              //设置 x 轴名称
set ylabel "y 轴"              //设置 y 轴名称
set grid                       //显示网格
set key left                   //将图例在图像左侧
```

通过上述设置，可以得到如图 8-2 所示图像。

如果想取消一个参数的设置，则可以通过 unset 命令来重置某个参数。另外，当不清楚某一条命令的具体用法时，可以灵活使用 help 命令来获得关于这个命令的提示和实例，如 help set xlabel。

2. 数据绘图

数据绘图是指利用具体的数据文件来绘制相应的图像。

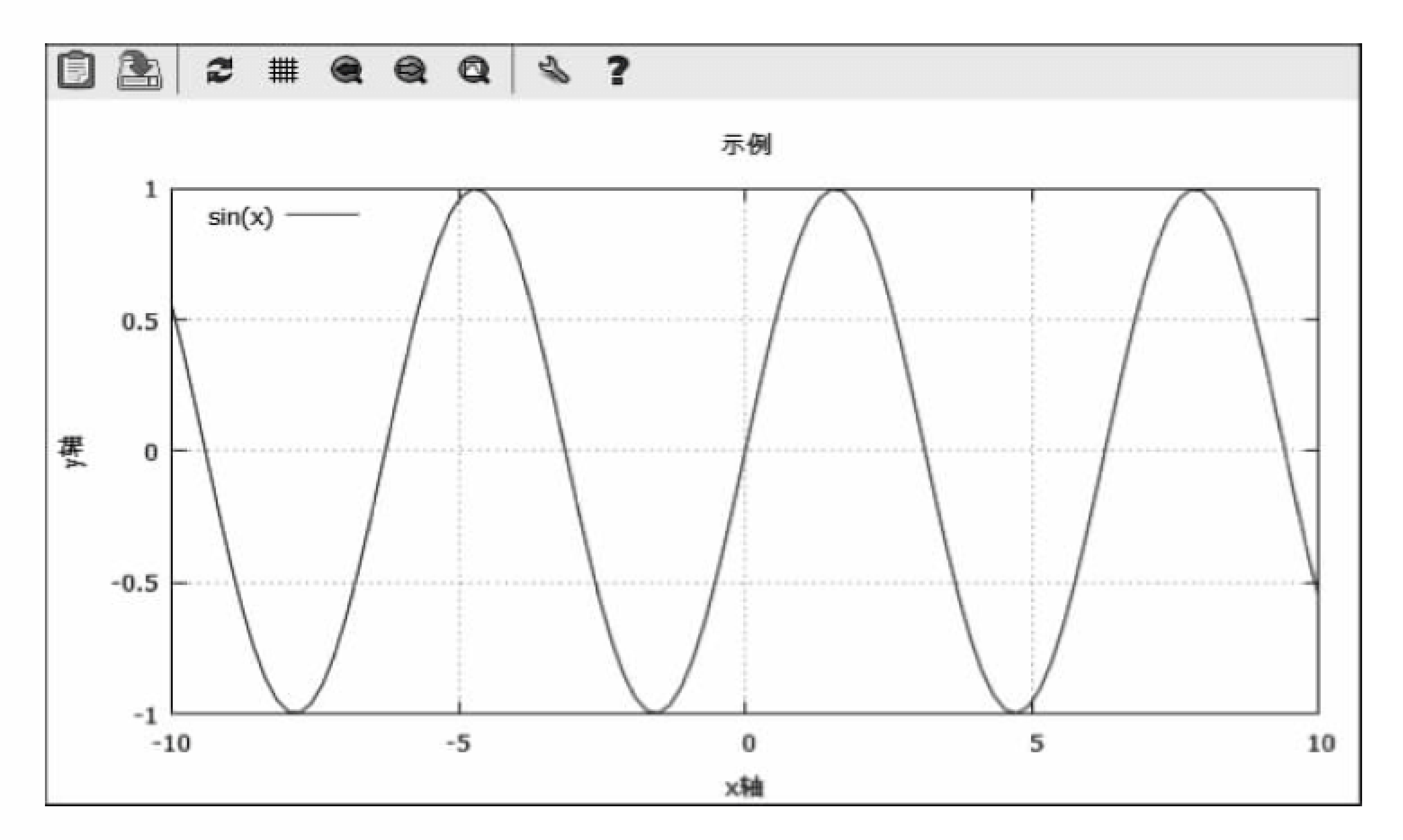


图 8-2　带参数的正弦曲线

例如，以纯文本的格式存储了数据文件 date. dat，它的内容如下：

```
# 月份　支出
# ============
1  21. 5
2  15. 1
3  10.2
4  25.4
5  27.9
6  27.1
7  15.3
8  12.9
9  28.3
10  17.8
11  23. 1
12  30. 5
```

在这个数据文件中，第一列为月份，第二列为当月的支出信息，# 之后是文件的注释信息，主要是帮助理解数据的内容。输入如下的命令：

```
set xlabel "月份"
set ylabel "支出"
unset key
set xrange [1:12]
set xtics 1,1,12
plot "data.dat" with lines
```

其中，with 命令表示图像绘制的方式。Gnuplot 支持大约 30 种的画图方式，points 方式是默认的画图方式。

结果如图 8-3 所示。

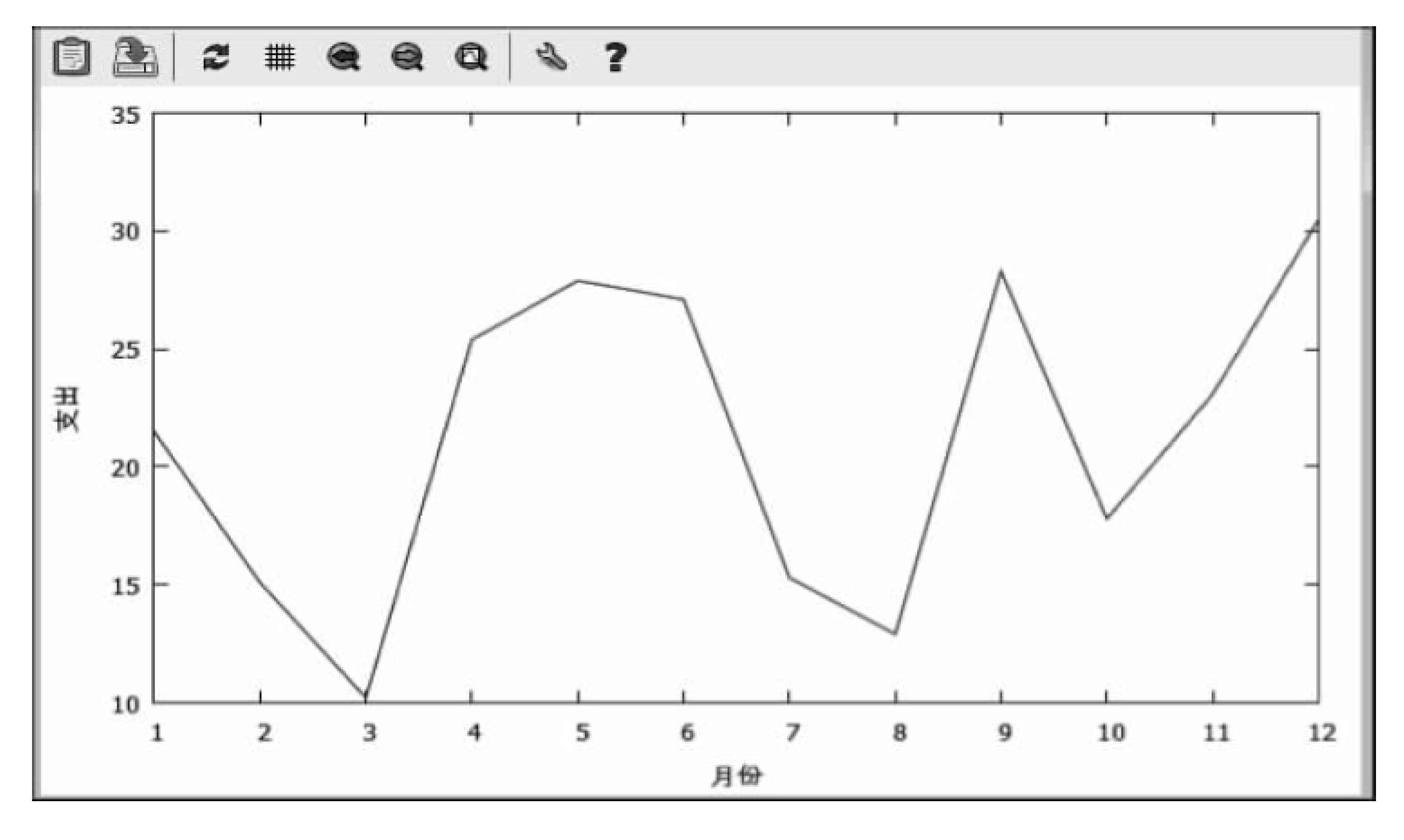


图 8-3 数据绘图图像

3. 同时绘制多条曲线

在实际工作中，经常会遇到要求绘制多组数据的情况，针对这种情况 Gnuplot 也提供了绘制多组数据的功能。本节中绘制的数据文件如下：

```
# 城市月平均降水量 (mm)
# 月份 北京 上海
# ======================
1  3.5  28.1
2  2.1  38.4
3  10.2  41.3
4  25.4  91.6
5  27.9  94.3
6  71.1  162.4
7  175.3  123.5
8  182.9  122.1
9  48.3  144.9
10  17.8  50.0
11  5.1  30.8
12  2.5  25.6
```

在 Gnuplot 中，执行下面的命令：

```
set xlabel "月份"
set ylabel "降水量"
set title "城市降水量分布"
set xrange [0.5:12.5]
set xtics 1,1,12
plot "data.dat" using 1:2 w lp pt 5,"data.dat" using 1:3 w lp pt 7
```

可以得到如图 8-4 所示图像。

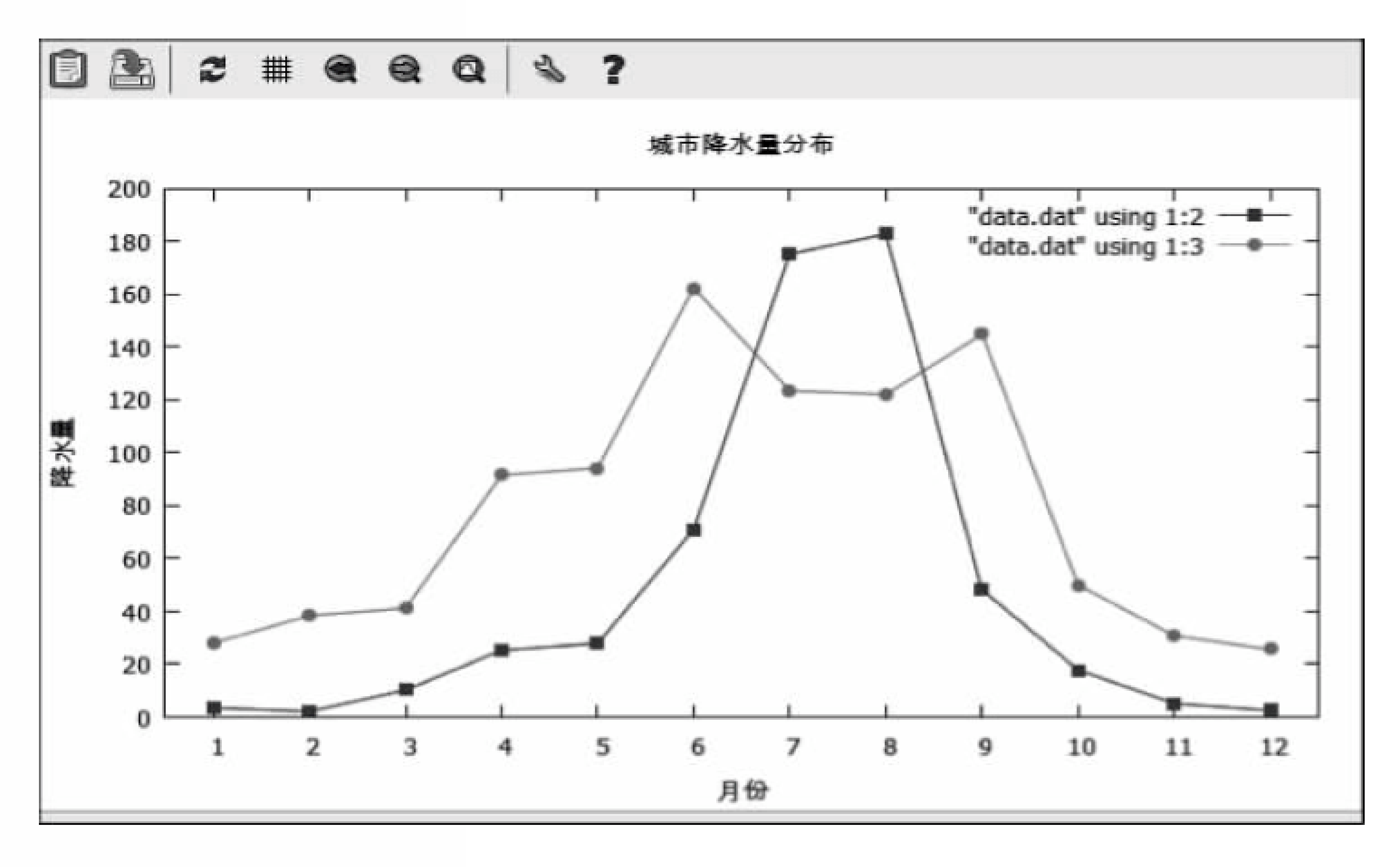


图 8-4　多组数据绘图

在这里使用到了命令 using，该命令的作用是告诉程序使用了哪一组数据进行绘图。例如，using 1:2 就是告诉程序使用第一列作为横轴，第二列作为纵轴进行绘图；using 1:3 是告诉程序使用数据的第一列作为横轴，第三列作为纵轴进行绘图。

4. 输出图片

到目前位置，对于生成的图片都是在屏幕上进行操作，下面将介绍如何将生成的图像进行保存。Gnuplot 提供了丰富的保存格式，如 png、pdf 等。其中，eps 是 latex 常用的图片格式，在这里我们就以 eps 格式为例讲解如何使用 Gnuplot 完成图像的输出任务。

首先对 Gnuplot 的终端进行设置：

```
set terminal postscript eps
```

接着可以执行下面的命令：

```
set xlabel "月份"
set ylabel "降水量"
set title "城市降水量分布"
set xrange [0.5:12.5]
set xtics 1,1,12
set output "data.eps"
plot "data.dat" using 1:2 w lp pt 5,"data.dat" using 1:3 w lp pt 7
set output
set term wxt
```

这里使用 set output 命令来指定输出文件的文件名。当上述命令执行结束后，就可以在当前的目录下找到输出的文件 data.eps。

8.2.3　程序调试工具

写程序的人都知道，Bug 是在所难免的，通过调试程序排除程序中的 Bug 是编程序

的一个非常重要的步骤。在整个程序开发周期中，调试所花的时间往往占据了极大的比例。采用什么样的调试技术、工具和手段往往决定了整个程序开发周期的长短以及最终程序的质量。所以调试技术也是非常值得重视的一个环节，本节就来讨论一下NS程序的调试技术。

NS程序的调试相对来说有不少困难，主要在于：

（1）没有针对OTCL程序的集成开发环境。在VC、VB、Delphi等集成开发环境中，程序可以随时运行调试，有单步运行、设断点、查看变量、查看内存等调试手段，很方便，功能很强大。但是要调试一个OTCL程序，就没有办法做到这样随心所欲。

（2）NS独特的分裂式的编程方法。NS使用者不仅要编写OTCL脚本，往往还要编写C++程序。NS的运行会通过OTCL的命令去执行C++程序的代码。这样，一旦一个程序的运行出了错误或者和预想的运行结果不一致，问题可能是出在OTCL脚本中，也可能是出在C++代码中。所以用户首先要判断Bug是在OTCL还是C++。这就给调试增加了很大的难度。

（3）NS的构件库非常庞杂。NS丰富的构件资源是它的优势所在，但另一方面这也增大了调试的难度。真正对用户有用的NS中的类很多都是从底层的基类开始经过许多次继承构造出来的。要确定一个成员变量或函数是这个子类的还是从基类继承的就得看源程序，在这些类之间查阅，难免会眼花缭乱。程序越复杂，调试越困难，这是很自然的。

（4）Linux环境的限制。NS是在Linux下开发的。虽然NS也提供Windows下的运行版本，但还是在Linux下的安装和使用更直接。然而，在Linux下开发可能有种种不便之处。首先，使用Linux以及在Linux下进行开发，在使用的工具、操作习惯等诸多方面都是和Windows有很大差别的。在中国，Linux的使用远远不如Windows普及，人们已经习惯了在Windows环境下工作和生活，对Linux不够适应。很多人觉得学习NS困难的一个很大的原因在于不适应Linux，不知道在Linux下面应该怎样做他们想做的事。其次，Linux下的一些工具确实还不如Windows下的工具使用起来方便，功能也没Windows的强大。例如，浏览、编辑和调试C++程序，用VC做这些事会让人感到很舒心，而如果在Linux下用vi去浏览编辑源代码，再用gcc编译，再用gdb调试，我想没人会说后者更方便。当然，Linux环境下已经有了集原程序浏览、编辑、编译、调试功能于一体的C++开发环境。随后就会讲解怎样利用这样的集成开发环境来调试NS程序。

下面将讨论如何调试NS程序的问题，包括如何调试OTCL程序以及如何调试C++部分的代码。

1. OTCL调试技术

1）利用输出提示信息调试

根据输出提示信息调试是最简单的调试技术，而且也是行之有效的。在脚本中加入一些输出语句可以帮助我们定位错误所在。在OTCL中，可以使用puts命令往控制台上输出一行字符串。OTCL是解释执行的语言，脚本修改后不必重新编译就可以执行，因此利用输出提示信息的方法调试OTCL脚本并不麻烦。

2）利用 OTCL 命令调试

OTCL 中的一些辅助命令也可以为调试提供帮助，例如 info 命令。利用 info 命令，可以查询对象并获取对象的当前状态信息。例如可以进行如下操作：

(1) info exists <变量名>——查询一个变量在当前作用域是否存在。

(2) <对象名> info class——查询一个对象的类名。

(3) <类名> info instances——查询一个类的所有实例。

(4) <类名> info instprocs——查询一个类的所有实例过程。

(5) <类/对象名> info vars——查询一个类/对象的成员变量。

(6) <类名> info superclass/subclass——查询一个类的超类/子类。

这些信息能够帮助了解当前程序的执行状态。

3）使用 TCL 调试器调试

必要时，也可以借助 TCL 调试器。目前，有两种 TCL 调试器可用：一个是 Karl Lehenbauer 的调试器，一个是 Don Libes 的调试器。NS 支持后者，因此下面的讨论中提到的都是这种调试器。

(1) TCL 调试器的安装。Don Libes 调试器可以从这里下载：http://expect.nist.gov/tcl-debug/tcl-debug.tar.gz。然后解包到和 ns-2.28 平行的目录。最后在安装目录中，执行以下命令：

```
$ ./configure
$ make
```

如果编译成功，将生成调试器的共享库。为了让 NS 中能够使用这个调试器，NS 编译时必须链接上这个库，因此必须把 NS 的 with-tcldebug 编译选项打开，并重新编译。在 ns-2.28 目录中操作如下：

```
$ ./configure -with -tcldebug
$ make clean
$ make
```

(2) 启动调试器。在 NS 中启动调试器的方法很简单，只要在脚本中任何希望开始调试的位置插入语句 debug 1 即可。例如，有脚本 test.tcl 如下：

```
puts "test debugging"
debug 1
proc myproc {x} {
   return [expr 2 * ( $ x + 1) ]
}
set a [expr 2 + 3]
set b "result is [expr [myproc $ a] + 1]"
puts $ b
```

执行该脚本，ns test.tcl 程序将在 debug 1 处暂停，并给出如下提示：

```
$ ns test.tcl
test debugging
2: lappend auto_path $ dbg_library
```

```
dbg2.0 > n
1: proc myproc {x} {
return [expr 2 * ( $ x + 1)]
}
set a [expr 2 + 3]
set b "resu…
```

这时，用户可以输入任何 NS 命令、OTCL 命令和调试器命令。在调试器提示符中，第 2 个数字表示 TCL 历史标识符，第 1 个数字表示当前调用栈的深度。每执行一步后，调试器都会显示下一步将要执行的命令。

常用的调试器命令：

(1) s——单步执行(进入过程)。

(2) n——单步执行(越过过程)。

(3) c——继续。

(4) r——继续执行直到从过程返回。

(5) u——转至上级作用域。

(6) d——转至下级作用域。

(7) w——列出调用栈("Where")。

(8) b——设置、清除或显示断点。

(9) h——帮助。

调试器命令的名字都是单个字母，并且和其他流行的调试器(如 gdb)的命令名很相似(甚至相同)。在调试状态下，用户除了可以执行调试命令外，任何 TCL 命令甚至是已定义的过程都可以执行。因此调试器命令中没有包含查看变量值的命令，用户可以使用 puts 等 TCL 命令来显示变量的值。

例如，在上面的例子中，可以通过 NS 命令进入过程 myproc 中，用 w 查看当前调用栈，然后用 puts 查看变量的值。为了在 myproc 执行体中能够查看外面的变量 a，先要使用 u 命令转至上级作用域。

2. gdb 调试技术

如果需要调试 C++代码，就需要使用 gdb。gdb 是 Linux 系统中包含的 GNU 调试程序，它是一个用来调试 C 和 C++程序的调试器。可以使程序开发者在程序运行时观察程序的内部结构和内存的使用情况。

gdb 所提供的一些功能如下：

(1) 运行程序，设置所有的能影响程序运行的参数和环境。

(2) 控制程序在指定的条件下停止运行。

(3) 当程序停止时，可以检查程序的状态。

(4) 修改程序的错误，并重新运行程序。

(5) 动态监视程序中变量的值。

(6) 可以单步执行代码，观察程序的运行状态。

gdb 程序调试的对象是可执行文件，而不是程序的源代码文件。然而，并不是所有的可执行文件都可以用 gdb 调试。如果要让产生的可执行文件可以用来调试，需在执行

gcc 指令编译程序时，加上-g 参数，指定程序在编译时包含调试信息。调试信息包含程序里的每个变量的类型和在可执行文件里的地址映射以及源代码的行号。gdb 利用这些信息使源代码和机器码相关联。

一般 Linux 系统都会自动装入 gdb。关于 xxgdb，mandarake linux 是默认安装，RedHat 的 6.2 版也是有的，后来不知道为什么没了。xxgdb 就是 gdb 上面封装了个图形界面，易用性有提高，尤其是设置断点，观察变量的值。灵活性打了折扣。但是对于 telnet 远程调试来说，xxgdb 还是不方便。据说 gdb 可以配合文本编辑器 emacs 使用（比 vi 功能更多的一种编辑器），但是没见到资料介绍。

gdb 有一个很强大的功能，就是连接到正在运行的程序上，而不一定非从 gdb 启动程序。

要使用 gdb 需要重新编译 NS。

（1）先备份原有工作：

```
tar -zcvf ns-2.1b9a.tar.gz ns-2.1b9a
```

（2）清除原有编译：

```
make clean
```

（3）修改 Makefile 文件，打开 debug：

```
DEFINE =  -g …
```

（4）重新编译：

```
make
```

运行时输入 gdb ns（或 ns 的可执行路径），出现如下提示符：（gdb）。这时可以输入 gdb 命令。gdb 命令的详细列表可以在 Linux 环境（注意并非 gdb 环境）下输入 man gdb 命令查看。man 命令是 manual（手册）的意思，是 Linux 通用的帮助文件格式。

在 gdb 环境下输入 help，将获得 gdb 的命令类别，然后输入 help 类别名来获得某个系列的命令名，输入 help 命令名可以获得命令的帮助。例如，输入 help running 可以获得运行系列的命令列表，如 attach、run 等。

例如这个 ns 环境：

（1）启动 ns 的 gdb 环境，如果是调试特定的 TCL 脚本，然后输入命令 set args simple-mipv6.tcl -mactrace OFF -NAM 1 -namfile .tmp/out.nam -stop 1000 -tracefile .tmp/out.tr。

（2）输入 b main，即在 main 函数处设置断点，b 文件名：行号。

（3）输入 run，程序就会在 main 处停下。

（4）输入 l，可以看到从 main 开始的 10 行程序。可以看到第一个函数是 Tcl_Main。

（5）设置断点 b Tcl_Main 或者 b 60，这是行号，在 list 的时候有输出。

（6）输入 s（表示 trace into，step into）或者 n（表示 step over，next），程序将在 Tcl_Main 处停止（因为这是第一行程序）。

（7）这时如果输入 list（就是 l 命令，也可以输入全称 list），可以看到这个函数的内

容。但是这个函数前面全是版权声明，只需要再直接按Enter键就可以重复上一条命令，再输入list就会显示下10条代码。

(8) 然后再输入s，跟踪进去，可以看到函数一行一行被执行的情况。

(9) 这时输入info variables可以看到所有这里有效的变量名，包括全局和局部的，这个命令执行很慢，在服务器上也要耐心等上1min。遗憾的是这里只有变量名没有变量值，而且由于ns的全局变量太多可以输入q打断输出。

这个命令调小程序可能有用，也就写在这里了。info指令可以提供不少信息，像info functions返回所有的可调用函数列表之类，不过都比较费时间。info locals据说可以记录很多输入信息。

(10) 观察一个变量的值可以使用三种方式：观察点和断点处就地打印，每次trace都打印一次。

设置观察点用watch命令，可以使某个变量只要一发生变化就停止程序。就地打印就很简单，prinf变量名就可以了。有时候莫名其妙打印不出来某个变量名，那是因为那个变量被gcc编译器优化掉了，没关系，相当于这里的代码没有被执行。每次trace都打印用的是display命令，只需要display这个变量，就可以每次（有效范围内的）单步调试显示这个变量的值。show values可以显示以前显示过的变量。每次打印一个变量，gdb都会分配一个编号，可以通过这个编号访问刚才的那个变量（如果变量名或表达式很长的话是有用的），如p $1。

(11) 不希望继续跟踪后，输入c、fg、continue三个命令中任何一个都可以继续运行。

(12) 上面仅仅是简单的使用。例如对我们有意义的，可以设置如下的断点：b MIPv6Agent::recv，这里只需要打上函数名就可以，不要参数和返回值类型之类。这个断点就比较有意义，可以通过其跟踪一些有意义的事情。不需要指定文件，除非函数名有冲突。解决的方法：'ipv6.cc':funcname。

(13) 要删除断点，需要在被断点拦住的时候（后面单跟的时候不算）输入clear，可以删除所有断点。输入info breakpoints可以看到所有的断点列表，每个断点有一个标号。单独删除这个断点是delete标号、del标号或者d标号（这个命令可以在任何gdb前缀时运行，不需要非得是被断点拦住的时候）。

总的来说，gdb的命令都有缩写，一般一两个字母。gdb支持Tab键命令提示和命令补全。支持上箭头重复上一条命令。连变量名/文件名/函数名都可以提示。例如p info，只需要输入p inf按Tab键，就会有所有inf开头的这里可能的变量名的提示。

如果要在gdb下运行shell程序，需要输入shell命令名，如shell ls、shell cd /home/等。

表8-1列出了一些gdb最常用的命令。

表8-1 gdb常用命令

命令缩写	命令全名	功能
l	list	列出代码行
b	break	设置断点
s	step	追踪

续表

命令缩写	命令全名	功　能
n	next	步进
c	continue	继续运行
r	run	运行应用
q	quit	退出 gdb
p	print	打印变量
/	watch	设置观察点
/	display	总显示某变量的值
d	del,delete	删除断点
/	info breakpoints	显示所有断点

8.3　支持 UMTS/WLAN 双模的网络仿真平台

8.3.1　UMTS 模块简介

UMTS 核心网的结构几乎和 GSM/GPRS 的相同，它几乎就是 GSM/GPRS 核心网的扩展来兼容新的 WCDMA 接入技术，所以需要在已有核心网基础上添加无线接入网模块以及用户终端模块，主要表现为与 NodeB 相关的模块以及与 UE 相关的模块。

为了搭建仿真场景，需要在现有的仿真平台上添加 UMTS 模块，这里选用的是 Strathclyde 大学提供的基于 ns-2.1b9a 的 UMTS 模块，可以在 http://www.geocities.com/opahostil/下载该模块，该模块只支持 UDP 数据源，有更好的 nam 视图，支持切换，添加了 UE 和 NodeB 两个新的节点，添加了新的协议层来处理 UTRAN 协议和功能，添加了 WCDMA 和新的路由模块，采用不分层地址，添加了系统动态更新路由表功能。

为了支持 TCP 数据源，还添加了 Hynderic Olivier and Raes Sven 修改的相关补丁，可以在 http://www.info.fundp.ac.be/~lsc/Research/ns-2_UMTS/下载。

UMTS 模块的特点是：

(1) 只有移动终端(UE)和基站(NODEB)。

(2) 没有 IP 地址，没有地址配置。

(3) 没有隧道机制，只可以完成简单的注册、切换、寻呼和资源分配。

(4) UE 没有双接口，只能在 UMTS 网内运动。

(5) 没有上层管理实体，难以进行移动性管理。

下面对该模块代码结构进行分析，主要列出了代码执行的主要部分，以分析 UMTS 中的工作原理。

1. UMTS 中的节点结构

UMTS 中的节点结构如图 8-5 所示。

根据图 8-5 中箭头的指向，可以在 NS 代码中确定对应的 target_、uptarget_和

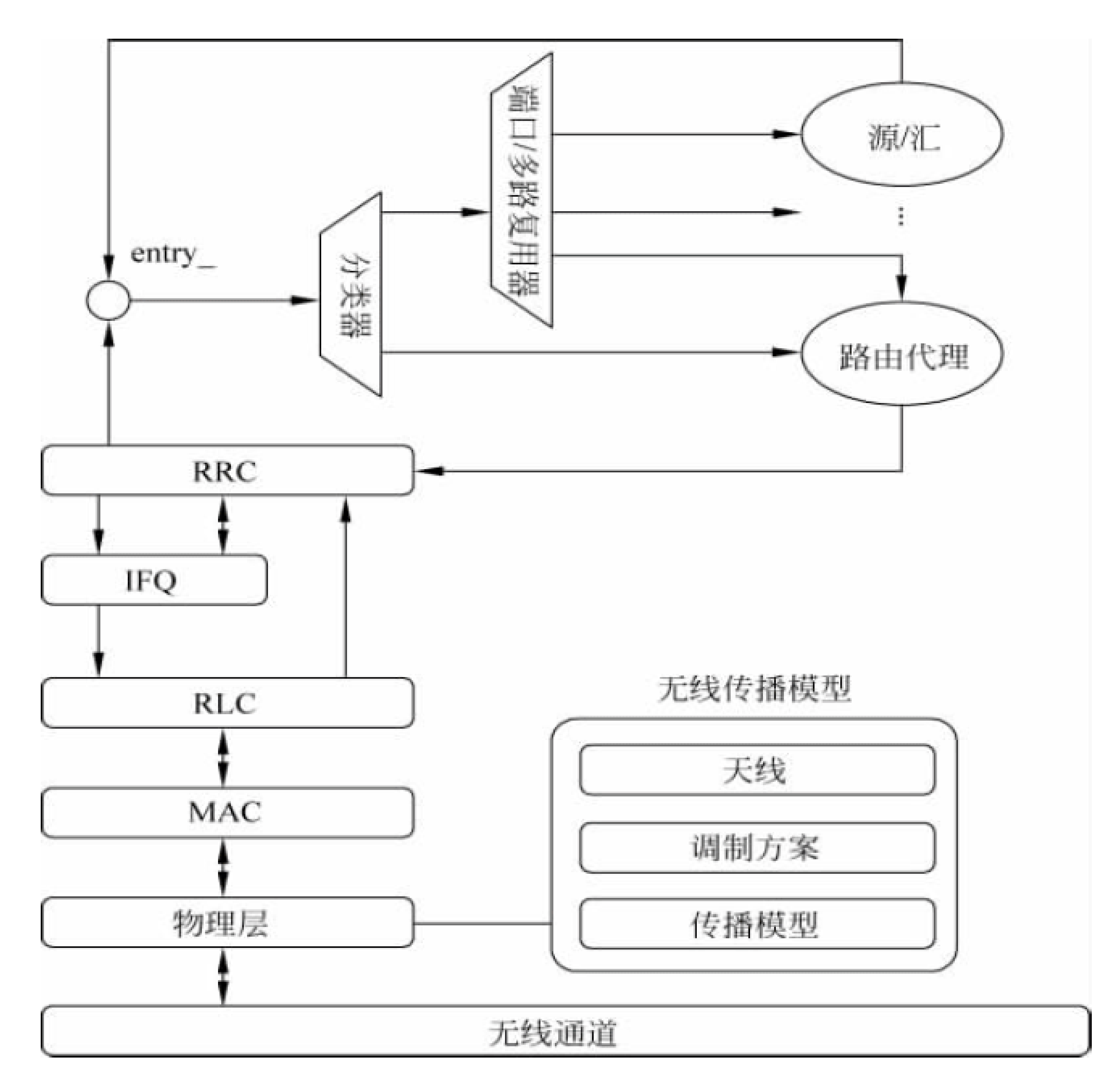


图 8-5 UMTS 中的节点结构

downtarget_(在 TCL 中设定)。图中节点结构的建立过程对应的 NS 代码是在 Node/MobileNode/UE instproc add-interface 函数中设置。

总体上来说,整个 UMTS 的 interface 的传送机制可以概括为信道的转换,每种不同的信令与数据都会有自己的逻辑信道,而这种逻辑信道的赋值与分类主要在 RRC 层进行,随即在 MAC 层进行逻辑信道与传输信道的转换,然后在物理层进行传输信道与物理信道的转换。物理层会根据传送的包所标注的相关物理信道设定不同的传输模型,将该包进行传输。

在此将针对整个节点结构对各个部分分别加以说明。

1) RRC 层实现的功能

RRC(radio resource control,无线资源控制)层基本上相当于原有的 LL 层。其功能为处理 UE 与 UTRAN 之间 L3 上的控制平面信令。RRC 层实现功能包括:广播由非接入层提供的信息,广播与接入层相关的信息,建立、维持及释放 UE 和 UTRAN 之间的一个 RRC 连接,建立、重配置及释放无线承载,分配、重配置及释放用于 RRC 连接的无线资源,RRC 连接移动功能管理,为高层 PDU 选路由,请求 QoS 的控制,UE 测量上报和报告控制,外环功率控制,加密控制,慢速动态信道分配,寻呼空闲模式下初始小区选择和重选,上行链路 DCH 上无线资源的仲裁,RRC 消息完整性保护和 CBS 控制。

在 NS-2 仿真代码中,RRC 表现为 UE 的 LLUE 实体,主要实现代码参考 umts/ll-ue.{cc,h},以及 NodeB 的 LLNodeb 实体,主要实现代码参考 umts/ll-nodeb.{cc,h}。为了和其他网络对象兼容,LLUE 和 LLNodeb 都是从 NS 中的 LL 类继承而来的。

2）IFQ 实现的功能

IFQ 就功能实现上来说是作为 RRC 的一部分存在的，它的主要作用是协助 RRC 层管理通信流并且控制传输速率。例如，在一个无线节点初次向基站申请与另外一个无线节点建立通信流时，基站在接受了申请之后会向目的节点发出寻呼请求，在接到无线节点的回复之前，源节点发送的数据包都会在 IFQ 中进行缓存。同时，在 UE 移动切换的过程中 IFQ 还可以将对应该 UE 的数据包提出，向上发送给 RRC 层，通过有线网络路由至 UE 的新地址，以此来完成转包机制。

在 NS-2 仿真代码中，IFQ 表现为 UE 中的 FCQueue 实体，主要实现代码参考 fc-queue.{cc，h}，以及 Nodeb 中的 BsFCQueue 实体，主要代码参考见 bsfc-queue.{cc，h}。FCQueue 和 BsFCQueue 都是从 NS 中的 DropTail 类继承而来的。

3）L2 层实现的功能

L2 层包括 MAC、RLC、PDCP、BMC 等 4 个子层，NS-2 仿真中只涉及 MAC 子层和 RLC 子层，实现功能分别为：

RLC 子层的主要功能是将上层的包进行分片以及重组，并根据分片的序列号管理数据流，不过，在实际的 NS-2 仿真过程中，机制略有不同，在接收分片的时候，只负责检查一个包的最后一个分片，如果可以收到最后一片，便认为整个数据包已经完整接受，然后更改最后一片的包大小的标识随即向上传送，对于可能存在的分片丢失的情况没有考虑。

在 NS-2 仿真代码中，RLC 子层表现为 UE 的 RlcUmts 实体，主要代码参考 umts/rlc-umts.{cc，h}，以及 NodeB 的 RlcUmtsNodeb 实体，主要代码参考 umts/ rlc--umts-nodeb.{cc，h}。

MAC 子层的功能包括：逻辑信道和传输信道之间的映射，为每个传输信道选择适当的传送格式，UE 数据流之间的优先级处理，UE 之间采用动态预调度方法的优先级处理，DSCH 和 FACH 上几个用户的数据流之间的优先级处理，公共传输信道上 UE 的标识，将高层 PDU 复接为通过传输信道传送给物理层的传送块，并将通过传输信道来自物理层的传送块复接为高层 PDU，业务量检测，动态传输信道类型切换，透明 RLC 加密，接入业务级别选择。

在 NS-2 仿真代码中，MAC 子层表现为 UE 的 MacUmts 实体，主要代码参考 umts/mac-umts.{cc，h}，以及 NodeB 的 MacUmtsNodeb 实体，主要代码参考 umts/mac-umts-nodeb.{cc，h}。

4）物理层实现的功能

在 OSI 参考模型中物理层（L1）处于底层，它提供物理介质中比特流传输所需要的所有功能。物理层的主要功能是完成传输信道与物理信道的映射，设定相关的传输模型，测量设定传输功率，并辅助完成一些切换功能。工作机制主要是采用时隙的方式，设定定时器，按照一定的规律定时接入 channel 传送数据，在传送一段时间之后断开连接，等待下一次的连接。数据包的传送频率等在 RRC 层进行分配，调制、编码等信息则携带在报头的某些特定域内进行传送。至于 WCDMA，也是采用时间槽的方式进行模拟的，具体可以分为两种结构，一种是固定的 10ms 帧，15 时隙结构，每个时隙 2560 个分片；另一种结构每个时隙的分片数由传输功率决定。

在 NS-2 仿真代码中，物理层表现为 UE 的 PhyUmts 实体，主要实现代码参考 umts/phy-umts.{cc,h}，以及 NodeB 的 PhyUmtsNodeb 实体，主要实现代码参考 umts/phy-umts-nodeb.{cc,h}。

2. UMTS 无线接口上的信道

在 UMTS 系统中移动用户终端 UE 与系统固定网络之间通过无线接口上的无线信道相连。这里介绍 UMTS 的无线接口，是为了说明各种通信信道，以便在后面介绍到的仿真中将信道的概念和 NS-2 代码相结合。UMTS 无线接口结构如图 8-6 所示，分为用户平面和控制平面，控制平面用于传输信令，用户平面用于传输用户数据。UMTS 总共有三种信道，它们分别是 RLC 层和 MAC 层之间通信的逻辑信道，MAC 层和 PHY 层之间通信的传输信道，以及不同物理层之间通信的物理信道。物理层提供不同的传输信道到 MAC 层，MAC 层通过不同逻辑信道给高层提供服务。传输信道特性由无线接口上传输信道物理特性进行描述，逻辑信道特性由传输消息的不同类型描述，物理信道特性在 FDD 制式中由码域、频率域确定。

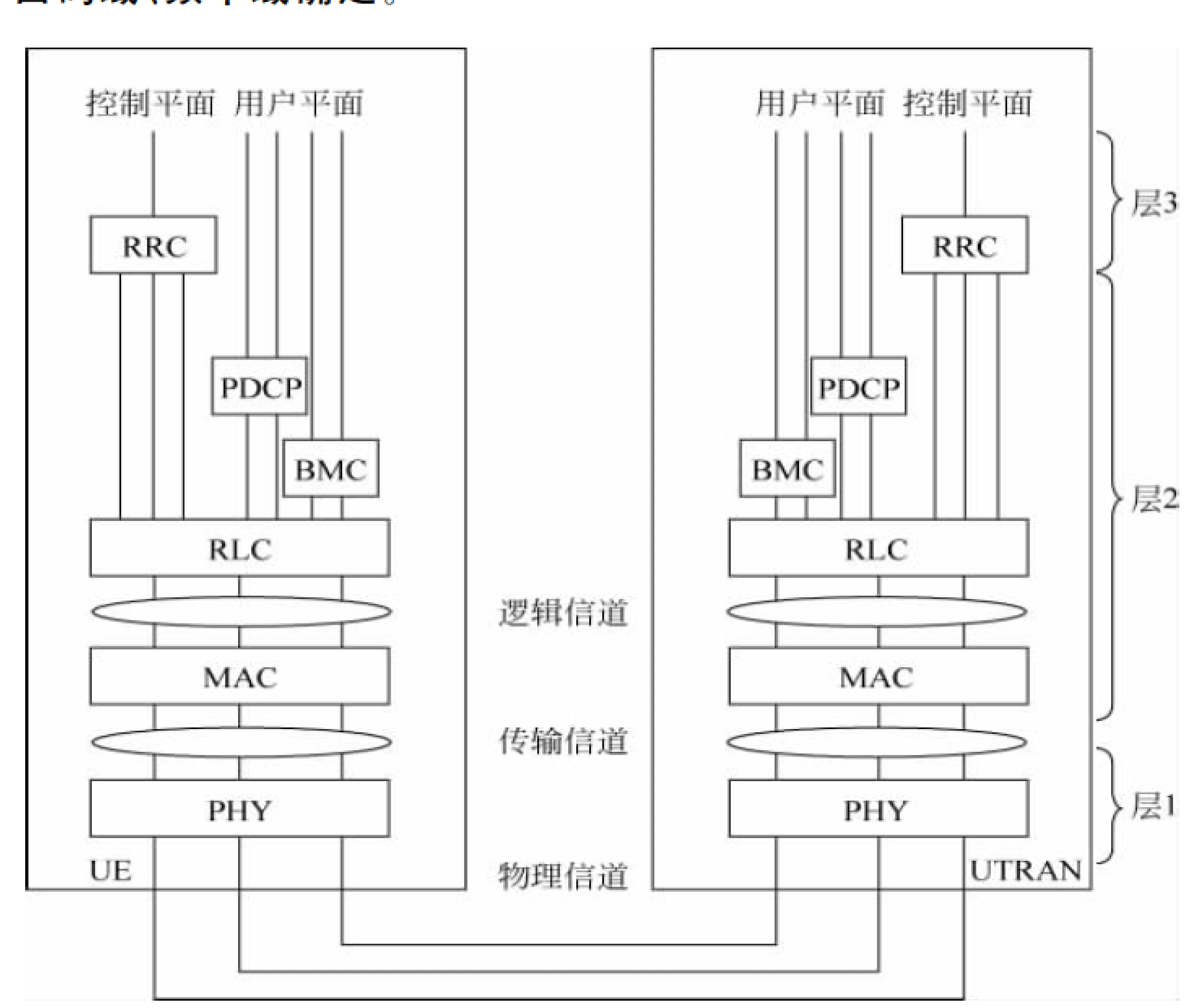


图 8-6 UMTS 接口结构

1）逻辑信道

不同的逻辑信道传输不同类型的信息，通常区分逻辑信道的方法是看它们传输的是用户平面的数据还是控制平面的数据。因此逻辑信道分为用户平面的业务逻辑信道和控制平面的控制逻辑信道。

（1）控制逻辑信道。为频分双工（FDD）模式设计的控制逻辑信道有以下种类：

① 广播控制信道（BCCH）——这个信道将控制信息发往蜂窝内的所有用户。

② 寻呼控制信道(PCCH)——这个信道用来通知一定范围内所有用户新接入的呼叫和信息。

③ 专用控制信道(DCCH)——这个信道用来给特定的UE传输信令。

④ 公共控制信道(CCCH)——这是一个在网络和UE间发送控制信息的双向信道,这个逻辑信道在上行链路中映射为RACH,在下行链路中映射为FACH。在后面的NS-2仿真过程中,信令大多数是在这个信道中收发。

(2) 业务逻辑信道。有两种已定义的业务逻辑信道:

① 专用业务信道(DTCH)——位于用户平面,用于给一个特定用户传输数据。

② 公共业务信道(CTCH)——用于给一定范围内所有UE发送专用用户信息。

2) 传输信道

UTRAN高层生成的数据在无线接口由传输信道承载,传输信道被定义为MAC层是如何通过无线接口传输数据的。一个既定的传输信道可以传输不同的逻辑信道传来的数据,而传输信道又在物理层上映射到不同的物理信道。传输信道有两种类型:专用传输信道和公共传输信道。

(1) 专用传输信道。专用传输信道仅仅为单个用户预留并在某个特定的频率采用特定编码加以识别。它只有一种类型,即专用信道(DCH)。专用信道用于发送既定用户物理层以上的所有信息。

(2) 公共传输信道。UTRAN FDD标准定义了6种公共传输信道:

① 广播信道(BCH)——这个信道传输来自广播逻辑信道的信息,发送UTRAN网络特定的信息或某一给定小区的特定信息给所有辖区内的UE,只存在于下行链路。

② 寻呼信道(PCH)——用于发送与寻呼过程相关数据的下行链路传输信道,也就是用于网络与终端开始通信时的初始化工作,和寻呼逻辑信道相对应。

③ 随机接入信道(RACH)——这是一个上行链路传输信道,用来发送来自终端的控制信息,如连接建立请求。

④ 前向接入信道(FACH)——这个信道是下行链路传输信道,用于向处于给定小区的终端发送控制信息,该信道用于NodeB收到随机接入请求之后。

⑤ 公共分组信道(CPCH)——这个信道存在于上行链路,是随机接入信道(RACH)的扩展,用来在上行链路方向发送基于分组方式的用户数据。

⑥ 下行链路共享信道(DSCH)——用来发送专用用户数据和控制信息的传输信道,它可以由几个用户共享。

3) 物理信道

物理信道由在上、下行链路中的无线信道组成,具体体现为不同的频段,物理信号则体现为电流脉冲。物理信道由专用物理信道和物理控制信道组成。

(1) 专用物理信道。

① 专用物理数据信道(DPDCH):分为上行和下行DPDCH。上行DPDCH用来承载由层2和更高层生成的专用数据。下行DPDCH传输真实的用户数据,并可以具有不同的扩频增益,比特速率是可以逐帧改变的。

② 专用物理控制信道(DPCCH):分为上行和下行DPCCH。上行DPCCH用来承

载在层 1 中产生的控制信息。下行 DPCCH 则传输包含已知导频信息、功率控制命令和一个可以选择的传输组合格式指示器组成的第一层产生的控制信息，其比特速率是恒定的。

（2）物理控制信道。

① 物理随机接入信道（PRACH）：属于上行信道，用于在上行链路中传送 RACH 传输信道。

② 物理共用分组信道（PCPCH）：属于上行信道，用于传送上行链路 CPCH 传输信道。

③ 基本公共控制物理信道（P-CCPCH）：用于下行链路传输 BCH 传输信道。它与 SCH 共用每一个时隙。每一个时隙的前 256 个码片用于 SCH。剩下的 2304 个码片用于主 CCPCH 来传输 BCH 传输信道。

④ 辅助公共控制物理信道（S-CCPCH）：用于在下行链路传输两个公共传输信道 FACH 和 PCH。FACH 和 PCH 能够共用一个二次 CCPCH，或者分别有自己的 S-CCPCH。S-CCPCH 传送的 PCH 必须能够覆盖整个小区，而不管物理信道传送的只是 PCH 还是 PCH 和 FACH。如果 S-CCPCH 只用于 FACH，那么就不必覆盖整个小区。

⑤ 同步信道（SCH）：由基站传送，在小区搜索过程中用于 UE。UE 为了重新得到由基站发送的广播信息，必须首先同基站正确同步。同步是 SCH 的首要条件。

物理信道除了有对应于上面介绍的传输信道之外，还有发送的信息只与物理层过程有关的信道。同步信道（SCH）、公共导频信道（CPICH）和捕获指示信道（AICH）对高层不是直接可见的，但从系统功能的观点来说，这些信道是必需的，每个基站都要发送这些信道。

3. 三层信道之间的映射关系

各层信道之间的映射关系会在随后提出的 NS-2 仿真代码中具体体现出来，信令会从 LL 层发出，经过各层信道发送和接收。具体映射关系如图 8-7 和图 8-8 所示。

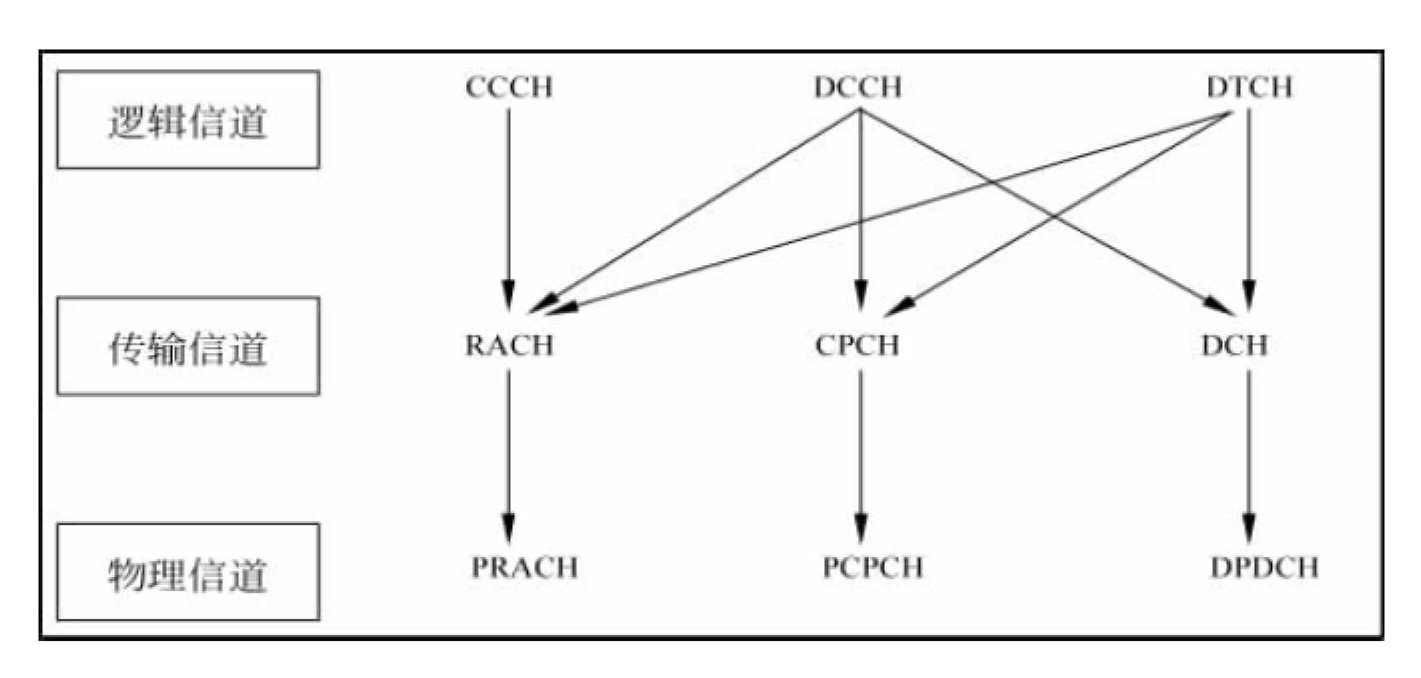


图 8-7　三层信道在上行链路中的映射关系

上面分析了无线结构中各层的结构以及主要的信道，下面将结合 UMTS 的基本通信过程来对 NS-2 代码做具体的分析。

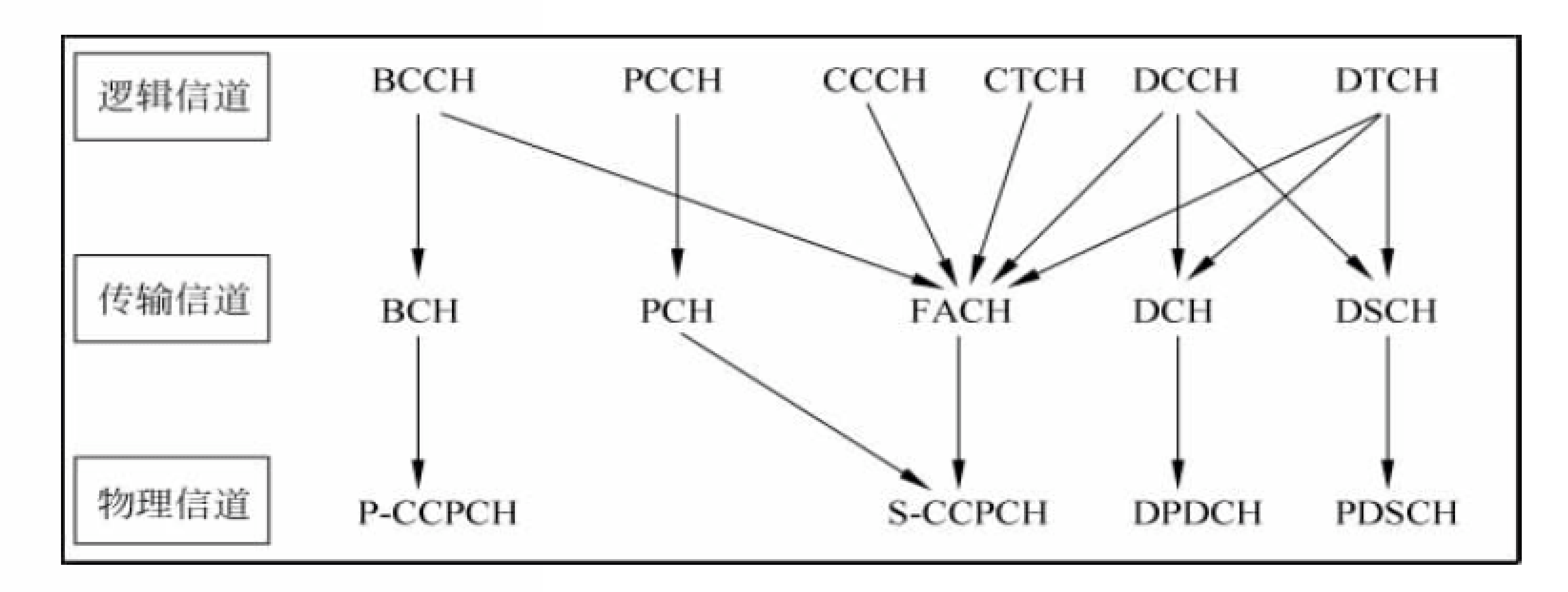


图 8-8　三层信道在下行链路中的映射关系

8.3.2　UMTS 基本通信过程及代码分析

UMTS 的基本通信过程包括小区搜索、手机注册、手机空闲、手机主叫、手机被叫、数据传输。下面通过物理信道的使用来了解整个过程。在后面的 NS-2 代码分析中除了说明下面的物理信道的信令流程，还将说明传输信道以及逻辑信道的使用情况。

1．UMTS 的基本通信过程分析

1）小区搜索过程

P-SCH ↓ => S-SCH ↓ => P-CPICH ↓ => P-CCPCH ↓ （注：↑ ↓ 分别表示上下行）

过程解释：

用户设备一开机，首先要寻找 NodeB，获得这个 NodeB 使用的主扰码，然后才能拿到小区开销信息。用户设备使用 P-SCH 信道主同步码（PSC）获得小区的时隙同步。然后再听辅同步信道上的辅同步码（SSC），判断当前扇区属于哪个主扰码组。接着，听主公共导频信道 P-CPICH，确定具体是哪个主扰码。

得到主扰码，就可以听到主公共控制物理信道 P-CCPCH，因为它也是用主扰码来加扰的，信道化编码固定（Cch，256，1）。它承载的内容是小区的系统消息。

2）用户设备注册/位置更新

PRACH Preamble ↑ => AICH ↓ => PRACH Message ↑ => S-CCPCH ↓ => DPDCH/DPCCH ↑ ↓

过程解释：

在上一过程中，用户设备了解了小区的情况，但是基站还不知道有移动台。所以，手机必须要有一个注册的过程。

UE 在初始接入时并不知道自己离基站有多远，所以在初始时 UE 不断在 PRACH 信道发随机接入前导（PRACH Preamble），试图先与基站系统取得联系，这是一个敲门的动作，同时也在进行开环功控。UE 会发送不止一个 Preamble，一开始做试探，功率小一点，看基站能不能听到。如果听不到，下一个 Preamble 的功率就增加一个步长，直到功率足够强，基站就听到了。Preamble 里的两个扰码（包括前导签名和前导扰码）来区分是哪个扇区哪个用户发来的前导。其中前导签名用来区分用户，前导扰码用来区分扇区。

基站听到后，通过捕获指示信道(AICH)告诉 UE 已收到呼叫请求前导，可以继续发送具体的接入请求信息。AICH 上的信息 AI 与 PRACH 上的签名对应。

于是，用户设备开始在 PRACH 上发送 PRACH Message。

基站收到用户设备的接入请求后，系统给 UE 分配资源，并通过辅公共控制信道 S-CCPCH 通知 UE(通过传输信道 FACH 来通知)，分配得到的主要是专用物理信道。

用户设备收到资源分配消息后，转到基站给它分配的 DPDCH/DPCCH 上进行注册或位置更新。这是双向的信道。

3) 用户设备空闲

P-CCPCH/PICH ↓

过程解释：

用户设备空闲时，要不断地监听主公共控制信道 P-CCPCH，在这上面经常会发送一些小区开销信息，如哪些状态发生改变。所以用户设备要想在这个小区继续工作，就要不断地了解小区的规则以及小区环境的变化。

用户设备还要监听寻呼指示信道，它会告诉用户设备在辅公共控制信道上有没有这个用户设备的寻呼消息。如果有，就转到辅公共控制信道上去收。

4) 用户设备主叫

PRACH Preamble ↑ => AICH ↓ => PRACH Message ↑ => S-CCPCH(FACH) ↓ => DPDCH/DPCCH ↑ ↓

过程解释：

用户设备发送接入前导，进行呼叫请求，开环功控；基站确认呼叫请求，发送 AI，通知用户设备继续发送具体接入请求；用户设备发送接入消息；基站通过 S-CCPCH (FACH)给用户设备分配信道；用户设备占用 PDCH 进行话音通信。

5) 用户设备被叫

PICH ↓ => S-CCPCH(PCH) ↓ => PRACH Preamble ↑ => AICH ↓ => PRACH Message ↑ => S-CCPCH(FACH) ↓ => DPDCH/DPCCH ↑ ↓

过程解释：

快速寻呼消息。通过监听 PICH，得到有给此用户设备的寻呼消息。这个消息不是具体的寻呼消息，具体的要到 S-CCPCH(PCH)中得到。

在 S-CCPCH(PCH)中得到寻呼消息后，用户设备就试图接入相应的基站，后面就和用户设备主叫的过程一样了。

6) 高速数据传输(上行)

CSICH ↓ => Access Preamble ↑ => AP-AICH ↓ => CD Preamble ↑ => CD/CA-ICH ↓ => PCPCH ↑

过程解释：

CSICH 指示一个 CPCH 信道的状态，即一个 CPCH 信道是不是可用。用户设备通过监听 CSICH，就可以知道有没有可用的 CPCH；如果用户设备知道有一个可用的 CPCH，就在 PCPCH 物理信道上发送接入前导。Preamble 同样进行敲门和开环功控；当基站收到请求，就通过接入前置捕获指示信道 AP-AICH 通知手机已收到请求。这是基站的第一次确认；用户设备收到第一次确认后，在 PCPCH 上发送碰撞检测前导 CD

Preamble，来检测碰撞；基站收到后，通过 CD/CA-ICH 来确认；用户设备收到第二次确认后，就开始在 PCPCH 上传送高速数据信息。

2. NS 代码分析

下面主要针对过程 1、2 和 3，即小区搜索过程、用户设备注册/位置更新过程，以及用户设备空闲过程对 NS 代码进行具体分析，由于篇幅限制，其他三个过程可以自己做类似的分析。为了简单起见，只写出每个函数中比较关键的步骤的代码，而且由于省略了一些括号，所以格式不一定正确，具体的代码请查看补丁包中的原始代码。

首先，用户设备 UE 开机：在 TCL 脚本中由 $ ue__ON 触发，ue__由 create-UE 命令建立。执行的代码在 mobilenode.cc 中：

```
MobileNode::command{
        if(strcmp(argv[1],"ON") == 0) {
            ll__->switch_on();              //ll_在初始化时设置为 LLUE
}
```

因此调用 ll-ue.cc 中的：

```
void LLUE::switch_on(void){
    send_setup();                           //发送接入请求
}
→
void LLUE::send_setup(void){
chsp->channel_t() = CCCH;                   //用于发送 SETUP 请求的逻辑信道为 CCCH
s.schedule(downtarget_,sp,delay_);
//LLUE 的 downtarget_为 RlcUmts,delay_为 2.5e-05s (ns-default.tcl 中设置)
}
```

在 RlcUmts 中未对 LL_SETUP 做处理，直接发送到 downtarget_，即 MacUmts：

```
void MacUmts::recv(Packet * p,Handler * h){
    switch (ch->channel_t()){
case CCCH:
            ch->channel_t() = RACH;         //CCCH 映射到传输信道 RACH
              downtarget__->recv(p,this     //MacUmts 的 downtarget_为 PhyUmts
}
→
void PhyUmts::recv(Packet * p,Handler * h){
   switch (ch->channel_t()) {
     case RACH:
          rach_recv__= p->copy();           //缓存该消息的副本，等有可用资源时发送
          if (lh->lltype() == LL_SETUP){    //说明是刚开机或者切换过程
          listen_sch__= 1;                  //持续监听 SCH 信道
          pCell__.start(20 * slot_time_);   //为小区搜索过程开启定时器 timer
}
```

其中，pCell__的定义为 CellTimer pCell__，超时后（slot__time__为 0.000667s，因此在 20 * 0.000667 = 0.01334s 后）将会调用：

```
void PhyUmts::cellHandler(){
    listen_sch_ = 0;                  //停止监听 SCH 信道
    if (selected_sc_< 0) {            //当前的 selected_sc_为初始值 -1
        Packet::free(rach_recv_);
        return;                       //没有选定的 scrambling code,释放 LL_SETUP 消息并返回
    }
    tx_preamble();                    //如果有选定的 scrambling code,则发送 RACH 前导消息
    return;
}
```

从上面的函数看出,只有当 selected_sc_被赋值后才会执行 tx_preamble()来发送 RACH 前导消息,那么 selected_sc_是什么时候赋值的呢?下面将分析 selected_sc_被设置的过程。

在 TCL 脚本中,NODEB 建立后,在构造函数 PhyUmtsNodeb::PhyUmtsNodeb()中有:

```
pDownSlot_.start((Packet *) (& intr_),0);
```

其中的 pDownSlot_是在 Phy-umts-nodeb.h 中定义的 DownSlotUmtsTimer 类型,超时时延为 0,因此立刻执行:

```
void DownSlotUmtsTimer::handle(Event * e){
    phy->downslotHandler(e);
}
→
void PhyUmtsNodeb::downslotHandler(Event * e){
pDownSlot_.start((Packet *)e,slot_time_);    //slot_time_(0.000667s)后再次超时
    mk_pccpch();                             //PCCPCH tx 主公共控制物理信道
    down_slot_++;
    if (down_slot_ == SLOTS_PER_FRAME) {     //SLOTS_PER_FRAME 定义为 15
        down_slot_ = 0;                      //ie (信息要素)的循环使用(wrap around)
    }
    if (down_slot_ == 0) {
        mk_sch();
//当前 down_slot_值为 1,而在以后 downslotHandler 再次超时后,将会执行 down_slot_++,即每
隔一个 slottime (0.000667s), down__slot__的值加 1,当到达 SLOTS__PER__FRAME,即过了
15 * 0.000667 = 0.010005s 时,将会执行 mk_sch();
}
```

在上面的 downslotHandler 函数中,将执行 mk_pccpch():

```
void PhyUmtsNodeb::mk_pccpch(void){
for (i = 0; i < MAX_NUM_SCRAM; i++){
        for (j = 0; j < MAX_NUM_SIG + 1; j++){
            phsp->free_resources_[i][j] = w_control.free_res_[i][j];
//广播上行可用的扰码(scrambling codes)和签名(signatures),free__resources__定义为: int
free_resources_[MAX_NUM_SCRAM][MAX_NUM_SIG + 1],是上行可用扰码和签名的矩阵,MAX_NUM__
SCRAM 为一个小区的最大扰码数,定义为 16
    }
for (i = 0; i < MAX_NUM_RACH; i++){
```

```
            phsp->rach_[i] = w_control.free_rach_[i];    //广播可用的 RACH 信道
    }
    phsp->sa_ = nodeb_address_;                 //携带自身 MAC 地址(物理地址)
    phsp->da_ = BROADCAST;                      //目的地址设置为广播地址
    chsp->channel_t() = PCCPCH;                 //PCCPCH 用于下行链路传输 BCH 传输信道
        pkttoTx_->enqueHead(sp->copy());        //将该数据包放入 pkttoTx_缓存队列中
    }
```

pkttoTx_队列中的数据包将在下次 pDownSlot_超时后通过空中接口发送出去，那么当 P-CCPCH 信道上的信号被 UE 接收到后：

```
void PhyUmts::recv(Packet * p,Handler * h){
if (umts_used_ == 1) {
//umts_used_初始值设置为 1,该值可能会在接口选择时发生变化
        if (ch->channel_t() == PCCPCH) {
                recv_from_phy(p,this);
}
→
void PhyUmts::recv_from_phy(Packet * p,Handler * h){
switch (ch->channel_t()) {
case PCCPCH:
if (ph->scrambling_c() == w_control.p_scrambling_) {
//由于当前的 w_control.p_scrambling_为 0,所以这次将不会进入这里
}
```

只有当 UE 的主扰码 w__control. p__scrambling__等于在 NodeB 中设置的 ph->scrambling__c()后才会执行 case P-CCPCH 中的代码。下面将分析 w__control. p__scrambling_的设置过程。

在前面提到，在 0. 010005s 后，downslotHandler 再次超时的时候，NodeB 将执行 mk_sch()：

```
void PhyUmtsNodeb::mk_sch(void){
chsp->channel_t() = SCH;                        //SCH 信道用于发送同步消息
    phsp->p_scrambling_c_ = w_control.p_scrambling_;        //发布该 Nodeb 的主扰码
    phsp->da_ = BROADCAST;
pkttoTx_->enque(sp->copy());                    //将该 SCH 数据包放入 pkttoTx_缓存队列中
}
```

pkttoTx_队列中的数据包将在下次 pDownSlot_超时后通过空中接口发送出去，那么当 UE 收到 SCH 信道上的消息后：

```
void PhyUmts::recv(Packet * p,Handler * h){
if ((ch->channel_t() == SCH) || (lh->lltype() == LL_UEAGENT)) {
            pRxPkt_.start(p,stime);
}
```

pRxPkt_定义为 RxPktUmtsTimer pRxPkt_，其超时后将调用：

```
void RxPktUmtsTimer::handle(Event * e){
    phy->recvHandler(e);
}
```

→
```
void PhyUmts::recvHandler(Event * e){
    recv_from_phy(p,this);
}
```
→
```
void PhyUmts::recv_from_phy(Packet * p,Handler * h){
  switch (ch->channel_t()) {
case SCH:
phyprdpl_->phypowerlist_->Insert(ph->rx_power(),ph->timestamp(),ph->sa());
                      //存储信号的接收功率,将在接口选择过程中用于比较小区的信号强度
if (listen_sch_&& (ph->p_scrambling_c() != w_control.p_scrambling_)) {
     //当前 UE 的 w_control.p_scrambling_为 -1,因此不等号成立
            cell_selection(p);                    //执行小区选择过程
}
```
→
```
void PhyUmts::cell_selection(Packet * p){
    if (ph->rx_power() > power_sch_){             //power_sch_初始值为 0,因此大于号成立
        selected_sc__ = ph->p_scrambling_c();     //存储主扰码
///此时 selected_sc_已经被赋值,注意查看前面提出的问题
        selected_p__ = ph->sa();                  //存储新的 NodeB 的物理地址
        power_sch__ = ph->rx_power();             //记录当前的接收功率
}
```

由于 selected_sc_已被赋值,则 0.01334s 后 pCell_超时后将执行:

```
void PhyUmts::cellHandler(){
    tx_preamble();                                //发送 RACH preamble
}
```
→
```
void PhyUmts::tx_preamble(){
//这里略去了一段代码,为查找可用的 RACH(w_control.rach_tx_)、扰码(w_control.scram_tx)以
//及签名(w_control.signature_)的过程
    //生成 preamble 消息
    Packet * sp = Packet::alloc();
    hdr_ll * lhrach_recv = HDR_LL(rach_recv_);
    hdr_ip * ihrach_recv = HDR_IP(rach_recv_);
    hdr_umts_setup * sesprach_recv = HDR_UMTS_SETUP(rach_recv_);
//注意这里 rach_recv_为前面的 LL_SETUP 时保存的 rach_recv__= p->copy();
if (lhrach_recv->lltype() == LL_SETUP){
        phsp->first_access__ = 1;            //标识为 UE"首次接入"到该 NodeB 中
        phsp->da__ = selected_p_;            //selected_p_的值被设置为 NodeB 的物理地址
}
    phsp->scrambling_c__ = index_c_;         //设置 UE 使用的扰码 scrambling code
    phsp->signature__ = w_control.signature_; //设置 UE 使用的签名 signature
    phsp->used_rach__ = w_control.rach_tx_;  //设置 UE 使用的 RACH
    chsp->ptype() = PT_PREAMBLE;             //消息类型设置
    chsp->channel_t() = PRACH;               //PRACH 在上行链路中传递 RACH
    rachtoTx_->enque(sp);                    //将 preamble 消息缓存到 RACH 传输队列中
}
```

要注意这个数据包是 PRACH 上的 PT_PREAMBLE 数据包,而不是前面保存的用

于 setup 请求的 LL_SETUP 数据包的副本，LL_SETUP 数据包的副本将在后面的步骤中被发出。

当 RachUpSlotUmtsTimer pRachUpSlot_超时后，将执行 rachupslotHandler，而在 UE 初始化时，PhyUmts 构造函数中有 pRachUpSlot_.start((Packet *)(& intr1_),0)，即最开始就超时一次，然后每次超时后，又有 pRachUpSlot_.start((Packet *)e，rach_slot_time_)，所以每隔 0.00125s 超时一次：

```
void PhyUmts::rachupslotHandler(Event * e){
pRachUpSlot_.start((Packet * )e,rach_slot_time_);    //重置 timer
if (rachtoTx_->length() > 0) {
//现在 rachtoTx_中存放的是 PT_PREAMBLE 数据包
tx_to_nodeb(prov);
}
→
void PhyUmts::tx_to_nodeb(Packet * p){
downtarget_->recv(p->copy(),this);        //将 PT_PREAMBLE 发送给 NodeB
}
```

当 NodeB 接收到 PT_PREAMBLE 数据包后，这时它首次从 UE 接收到信号：

```
void PhyUmtsNodeb::recv(Packet * p,Handler * h){
if (nodeb_address_ == ph->da()) {
//已知 ph->da_被设置为本 NodeB 的物理地址
        if ( packet_for_me(p) ) {         //查看该 UE 是否已经注册到本 NodeB 中
                pRxPkt_.start(p,stime);
}
→
int PhyUmtsNodeb::packet_for_me(Packet * p){
    i = look_for(ph->sa());                //在 addresses table 中查找该 UE 的信息
    if (i != -1){                          //该 UE 已经注册到了本 NodeB 中
        ph->sa() = i;
        return (1);
//由于 UE 是首次接入，不会进入这里，因此不执行 return，而是继续向下执行
    }
    if (ph->first_access_ == 1){           //已知该数据包的 ph->first_access_被设为 1
        //如果该用户是第一次接入：执行 setup 或者 handover
        for (i = 0; i < MAX_NUM_UE; i++){
         if (ue_id[i] < 0){
         ue_id[i] = ph->sa();              //将 UE 注册到 NodeB 物理层 addresses table 中
            ph->sa() = i;                  //将 UE 的物理地址改变为内部地址
return (1);
}
```

因此，UE 先在物理层进行注册，然后继续往上层传送数据包，注意此时 ph-> sa() 已被改变。

由于返回值为 1，则还将执行 pRxPkt_.start(p，stime)，pRxPkt_超时后调用：

```
void PhyUmtsNodeb::recvHandler(Event * e){
    recv_from_phy(p,this);}
→
```

```
void PhyUmtsNodeb::recv_from_phy(Packet * p,Handler * h){
switch (ch->channel_t()) {
    case PRACH:
    //在 PRACH 信道上发送的可能是 RACH preamble 或者 RACH message
        if (ch->ptype_ == PT_PREAMBLE){  //收到的是 RACH preamble
         if (w_control.free_res_[ph->scrambling_c_][ph->signature_] == 0) {
         //preamble 中扰码所对应的签名 signature 可用(即尚未被使用)
            sig_ue_[ph->sa()] = ph->signature_;
            tx_aich(p);                   //发送 AICH 消息,通知 UE 可发送 RACH message
}
→
void PhyUmtsNodeb::tx_aich(Packet * p){
chsp->channel_t() = AICH;
//AICH 属于公共指示信道,用于给 UE 提供上行接入捕获指示 AI(Acquisition Indicator),通知其
//接入信息已被系统获知,该信息与 PRACH 中的 Signature 相对应. 在 AICH 上传送系统对接入信
//息已被捕获的确认消息
aichtoTx_->enque(sp->copy());
}
```

当 NodeB 的 AichSlotUmtsTimer pAichSlot_超时后(在构造函数中有 pAichSlot_.start((Packet *)(& intr1_),0),所以在一开始就超时一次,且以后每次超后都调用 pAichSlot_.start((Packet *)e,aich_slot_time_)):

```
void PhyUmtsNodeb::aichslotHandler(Event * e){
    pAichSlot_.start((Packet *)e,aich_slot_time_);
    if (aichtoTx_->length() > 0) {
        radioSwitch(ON);
        tx_to_ue(aichtoTx_->deque());
}
```

AICH 信道上的消息被 UE 接收,在 RxPktUmtsTimer pRxPkt_超时后执行:

```
void PhyUmts::recv_from_phy(Packet * p,Handler * h){
    switch (ch->channel_t()) {
        case AICH:
            if ((ph->scrambling_c() == w_control.scram_tx_) && (ph->signature() ==
w_control.signature_) && (ph->da() == ue_address_)) {
        //收到 NodeB 对 RACH Preamble 的响应,说明现在可以发送 RACH message 了
        mk_msg_rach();                          //发送 RACH 的 message 部分
}
→
void PhyUmts::mk_msg_rach(){
   hdr_ll * lhsp = HDR_LL(rach_recv_);          //rach_recv_为最初 LL_SETUP 的副本
if (lhsp->lltype() == LL_SETUP){
        phsp->da() = selected_p_;               //selected_p_为 NodeB 的物理地址
        chsp->next_hop() = BROADCAST;
}
chsp->channel_t() = PRACH;                      //RACH message 消息在 PRACH 信道上发送
rachtoTx_->enque(rach_recv_->copy());           //保存到 rachtoTx_缓存队列中
}
```

下一次 RachUpSlotUmtsTimer pRachUpSlot_超时后，将向 NodeB 发送该 LL_SETUP 数据包，当 NodeB 接收到之后：

```
void PhyUmtsNodeb::recv(Packet * p,Handler * h){
if (nodeb_address_ == ph->da()) {
        if (packet_for_me(p)) {
                pRxPkt_.start(p,stime);
}
```

packet_for_me 函数的过程前面已经介绍过，在 RxPktUmtsTimer pRxPkt_超时后：

```
void PhyUmts::recv_from_phy(Packet * p,Handler * h){
    switch (ch->channel_t()) {
        case PRACH:
            if (ch->ptype_ == PT_PREAMBLE){   //这次不是 PT_PREAMBLE
            …
} else { //接收到的是 RACH message
                ch->channel_t() = RACH;       //PRACH 映射到 RACH
                ph->sa_ = ue_id[ph->sa()]; //将物理地址变为内部地址
                uptarget_->recv(p,this);      //将 RACH message 发送到 MAC 层
}
→
void MacUmtsNodeb::recv(Packet * p,Handler * h){
switch (ch->channel_t()) {
    case RACH:
        ch->channel_t() = CCCH;               //RACH 映射到 CCCH 逻辑信道
        uptarget_->recv(p,this);              //将 CCCH 发送到 LL 层
}
→
void LLNodeb::recv(Packet * p,Handler * h){
    switch (ch->channel_t()){
     case CCCH:
        if (lh->lltype() == LL_SETUP) {
         if (register_ue(p)) {                //在 RRC 层注册该 UE，保持相关信息
              send_setup_reply(p);            //向 UE 发送对 LL_SETUP 请求的回复
}
→
void LLNodeb::send_setup_reply(Packet * p){
    lhsp->lltype() = LL_SETUP_REPLY;          //回复 LL_SETUP_REPLY 消息
    chsp->channel_t() = CCCH;
    s.schedule(downtarget_,sp,delay_);
//delay_时间后，将由 downtarget_( RlcUmtsNodeb)接收
}
```

随后将是 RlcUmtsNodeb 发送到 MacUmtsNodeb，然后 MacUmtsNodeb 将逻辑信道 CCCH 映射到传输信道 FACH，并发送到 PhyUmtsNodeb。物理层接收后：

```
void PhyUmtsNodeb::recv(Packet * p,Handler * h){
switch (ch->channel_t()) {
        case FACH:
        //FACH 信道一般用于对 setup 和 handover 请求的应答、资源分配等
```

```
                if (lh->lltype() == LL_SETUP_REPLY) {
                    //执行 setup 或者 handover 过程
                    ue_registry(p);             //在 NodeB 的物理层注册新的 UE
                }
                tx_fach(p);                     //发送 FACH message
}
→
void PhyUmtsNodeb::tx_fach(Packet * p){
    if (lh->lltype() == LL_SETUP_REPLY) {
        //在前面的 ue_registry 函数中给 UE 分配了寻呼组(paging group)
        i = look_for(ph->da());
        ph->paging_group_ = ue_info_[i].paging_group;     //设置 UE 的寻呼组
    }
    ch->channel_t() = SCCPCH;                   //FACH 映射到物理信道 S-CCPCH
    pkttoTx_->enque(p->copy());                 //将数据包放入在 pkttoTx_缓存队列中
}
```

当 DownSlotUmtsTimer pDownSlot_超时后将执行：

```
void PhyUmtsNodeb::downslotHandler(Event * e){
while (pkttoTx_->length() > 0) {
tx_to_ue(pkttoTx_->deque());                    //发送刚才缓存的 S-CCPCH 信道上的消息
}
```

当 UE 收到后该 LL_SETUP_REPLY 消息后：

```
void PhyUmts::recv_from_phy(Packet * p,Handler * h){
switch (ch->channel_t()) {
     case SCCPCH:
        if (((ph->scrambling_c() == w_control.p_scrambling_) || (ph->scrambling_c()
== selected_sc_)) && ((ph->da() == ue_address_) || (ph->da() == ip_ue_))){
          //接收到 NodeB 对 LL_SETUP 请求的回复消息
          if (lh->lltype() == LL_SETUP_REPLY){
            w_control.p_scrambling_ = selected_sc_;     //注意：UE 的主扰码被设置
            ch->channel_t() = FACH;             //SCCPCH 映射到 FACH
            uptarget_->recv(p,this);            //向 uptarget_(RlcUmts)发送，进行后续处理
}
```

RlcUmts::recv 函数中直接将数据包发送到 uptarget_(MacUmts)：

```
void MacUmts::recv(Packet * p,Handler * h){
switch (ch->channel_t()){
    case FACH:
        ch->channel_t() = CCCH;                 //FACH 映射到 CCCH
        uptarget_->recv(p,this);                //向 uptarget_(LLUE)发送
}
→
void LLUE::recv(Packet * p,Handler * /* h */){
if (lh->lltype() == LL_SETUP_REPLY) {
        ue_state_ = 2;           //改变 RRC 层 UE 状态：0: OFF,1: waiting to turn on,2: ON
        phy_->ue_state_ = 2;                    //改变物理层的 UE 状态
        ue_address_ = phy_->ue_address_;        //记录 UE 的物理地址(MAC 地址)
```

```
        MobileNode * node__ = (MobileNode * )(phy__->netif__->node());
        ip_ue__ = node__->address();
        phy__->ip_ue__ = ip_ue_;
        ifq__->ip_ue__ = ip_ue_;
        rlc__->addr() = ip_ue_;
        //上面三个操作是为了在物理层、Ifq、Rlc 层都可读取本节点的 IP 地址
        nodeb_address__ = phy__->nodeb_address_; //记录当前 NodeB 的物理地址
        ip_nodeb__ = ih->saddr();               //记录当前 NodeB 的 IP 地址
}
```

由于此时 UE 的主扰码 w_control. p_scrambling_已经设置(在刚才的 PhyUmts::recv_from_phy 函数中),UE 得到主扰码,就可以听到主公共控制物理信道 P-CCPCH,因为它也是用主扰码来加扰的。那么下次 P-CCPCH 信道上的信号被 UE 接收到后:

```
void PhyUmts::recv_from_phy(Packet * p,Handler * h){
switch (ch->channel_t()) {
case PCCPCH:
    if (ph->scrambling_c() == w_control.p_scrambling_) {
    if ((ph->rx_power() <= threshold_) && (ph->rx_power() < power_sch_) && (!handover
_)) {                                           //没有收到信号更强的其他小区的消息,
所以暂时不进入这里
     …
    }
    power_sch__ = ph->rx_power();               //UE 更新当前的接收功率
    for (i = 0; i < MAX_NUM_SCRAM; i++){        //UE 更新可用资源
        for (j = 0; j < MAX_NUM_SIG + 1; j++){
            w_control.free_res_[i][j] = ph->free_resources_[i][j];
        }
    }
    for (i = 0; i < MAX_NUM_RACH; i++){         //UE 更新可用的 RACH
        w_control.free_rach_[i] = ph->rach_[i];
    }
```

UE 收到资源分配消息后,转到 NodeB 给它分配的 DPDCH/DPCCH 上进行注册或位置更新。在 UMTS 模块补丁中,没有关于注册和位置更新的代码,是在双模模块中才加入的。后面介绍双模模块时将介绍这些过程。

当 UE 成功注册到 NodeB 中且处于空闲状态时,要不断地监听主公共控制信道 P-CCPCH,在这上面经常会发送一些关于 NodeB 的开销信息,如哪些状态发生改变等。所以手机要想在这个小区生活下去,就要不断地了解 NodeB 的规则以及环境的变化,且要监听接收功率,如果接收功率低于接收阈值,则要执行切换过程。此外,UE 还要监听寻呼指示信道 AICH,它会告诉 UE 在辅公共控制信道上有没有该 UE 的寻呼消息。如果有,就转到辅公共控制信道上去收。

8.3.3 NS-2 的无线模块简介

NS 的无线模块最初是由 CMU 的 Monarch 工作组引入到 NS 中的。在本节中将介绍 MobileNode(移动节点)的工作机制、路由机制和用来构造 MobileNode 的网络协议栈

的各个网络构件。这些网络构件包括 Channel（信道）、Network Interface（网络接口）、Radio Propagation Model（无线信号传输模块）、MAC 协议、Interface Queue（接口队列）、Link Layer（链路层）和 ARP（地址解析协议）等。

由 CMN/Monarch 引入的无线模块可以进行纯无线网络（包括无线局域网 WLAN 和多跳的 ad-hoc 网络）的模拟，进一步扩展后的无线模块还支持无线和有线网络联合模拟以及移动 IP（Mobile IP）模拟。

无线模块是以 MobileNode 为基本核心的，并添加一些新的特性来支持多跳 ad-hoc 网络和无线局域网的模拟。C++ 的 MobileNode 类是 Node 类的派生类。MobileNode 由基本的 Node 再加上无线和移动节点所需要的功能（如在给定的拓扑中移动，通过无线信道接收和发送信号等来创建移动、无线的模拟环境）。它们之间一个主要的区别是 MobileNode 不会通过 Link 连接到其他的 Node 或 MobileNode。

MobileNode 是一个由 C++ 和 OTCL 共同实现的对象，它的功能（包括节点移动、周期性的位置更新、维护拓扑边界等）是在 C++ 中实现的，而设定 MobileNode 的各个网络构件（如 Classifier、LL、MAC、Channel 等）则是在 OTCL 中实现的。

目前 MobileNode 所支持的 ad-hoc 路由协议主要包括 DSDV（destination sequence distance vector）、DSR（dynamic source routing）、TORA（temporally ordered routing algorithm）和 AODV（adhoc on-demand distance vector）。下面介绍在 TCL 中创建一个 MobileNode 的方法：

```
 $ ns_node - config -adhocRouting $ opt(adhocRouting) \
-llType $ opt(ll) \
-macType $ opt(mac) \
-ifqType $ opt(ifq) \
-ifqLen $ opt(ifqlen) \
-antType $ opt(ant) \
-propInstance [new $ opt(prop)] \
-phyType $ opt(netif) \
-channel $ opt(chan) \
-topoInstance $ topo \
-wiredRouting OFF \
-agentTrace ON \
-routerTrace OFF \
-macTrace OFF
```

上面这个 node-config 函数用来配置一个移动节点，可以配置的选项包括 ad-hoc 路由协议、网络协议栈、信道、拓扑、无线传输模型，以及是否打开有线路由、是否打开各层（agent、router、mac）的 trace。如果使用了 hierarchical 地址，还需要提供节点的 hier 地址。

通过下面的方法来真正实现创建移动节点：

```
for {set i 0} { $ i < $ opt(nn)} {incr i} {
  set node_( $ i) [ $ ns_node]
}
```

上面的过程创建了一系列的 MobileNode 对象，同时根据在 node-config 中的设置，

为每个 MobileNode 对象创建了指定的 ad-hoc 路由协议和网络协议栈，并将协议栈中的各个构件互联并连接到信道上。MobileNode 的结构如图 8-9 所示。

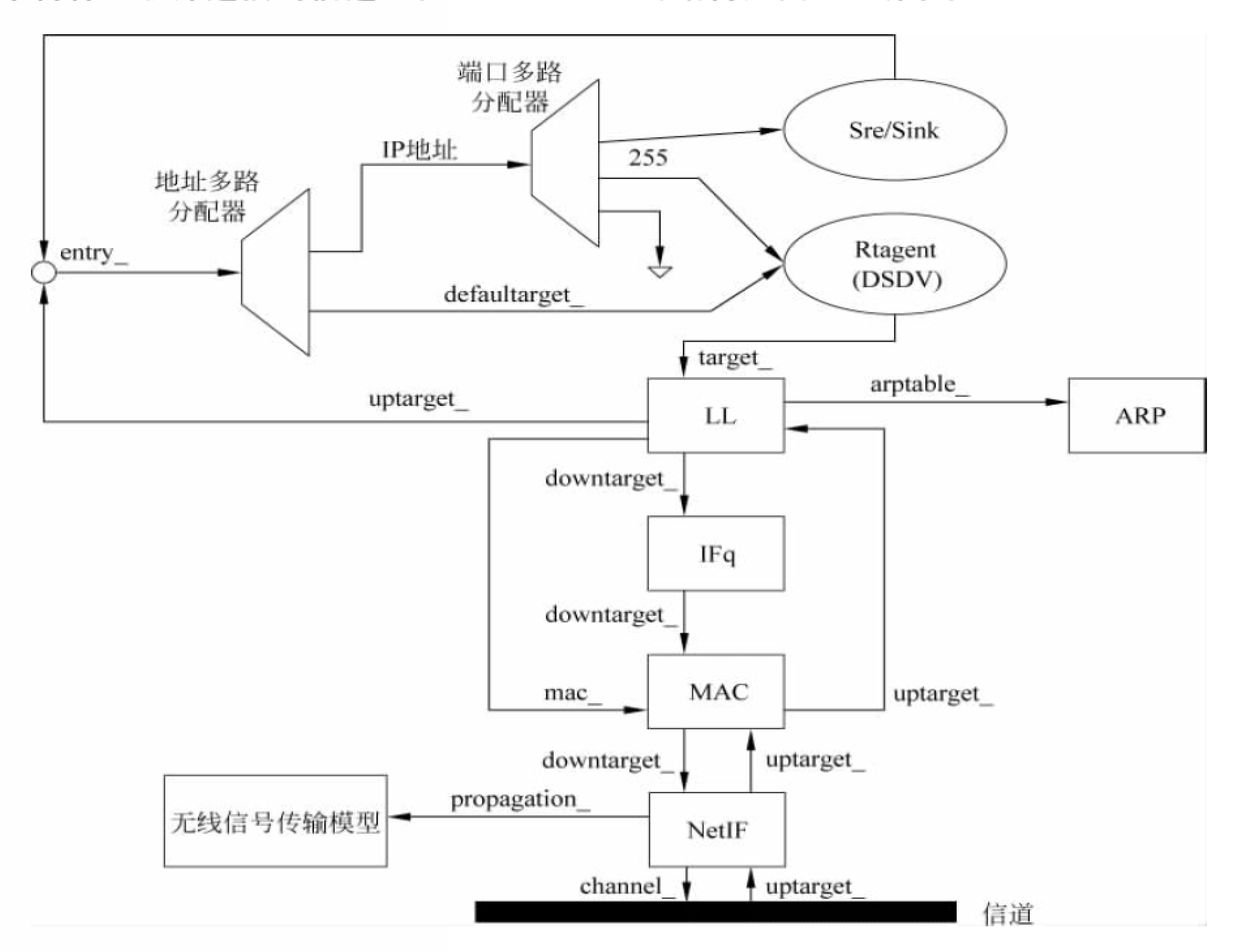


图 8-9　MobileNode 的结构示意图

1. 移动节点的组成

这部分将介绍移动节点的基本工作机制。移动节点是由一系列的网络构件构成的，这些构件包括链路层(link layer，LL)、连接到 LL 上的 ARP(address resolution protocol)模块、接口队列(interface queue，IFq)、MAC 层(MAC)、网络接口(network interface，NetIF)。移动节点通过网络接口连接到无线信道上。移动节点的各个网络构件是在 OTCL 中创建并合并在一起的，NS-2 中相关代码为 MobileNode 类的 add-interface 方法：

下面将简单介绍移动节点的各个网络构件以及无线信道。

(1) LL：移动节点使用的 LL 与 NS 手册的第 14 章介绍的 LL 基本相同。唯一的区别是移动节点的 LL 连接了一个 ARP 模块，用来把 IP 地址解析成物理(MAC)地址。通常，对于所有发出的分组，路由 agent 会把分组传递给 LL。LL 把分组传递给接口队列。对于所有接收到的分组，MAC 层将分组传递给 LL，LL 再将分组传递给 node_entry_。NS-2 中 LL 类的代码位于～ns/mac/ll.{cc，h}以及～ns/tcl/lan/ns-ll.tcl。

(2) ARP：地址解析协议模块从 LL 接收请求。如果 ARP 已经知道了目标节点的物理(MAC)地址，它就把该物理地址写入分组的 MAC 头中。否则，它就广播一个 ARP

请求并暂时缓存当前的分组。对于每一个未知的目标物理地址，都有一个可以存放一个分组的缓冲区。当更多的传送给同一个目标节点的分组被送到 ARP 模块时，前面被缓冲的分组就被丢弃掉。一旦 ARP 知道了分组的下一跳目标节点的物理地址，该分组就被放入接口队列中。ARPTable 类的代码位于～ns/mac/arp.{cc，h}以及～ns/tcl/lib/ns-mobilenode.tcl。

(3) IFq：接口队列是由 PriQueue 类引起的，PriQueue 类是一个优先级队列，它优先处理路由协议分组。它可以对所有队列中的分组进行过滤，删除那些具有特定目标地址的分组。PriQueue 类的代码位于～ns/queue/priqueue.{cc，h}。

(4) MAC 层：MAC 层实现了 IEEE 802.11 的 DCF(distributed coordination function)MAC 协议。Mac802_11 类的代码位于～ns/mac/mac-802_11.{cc，h}。

(5) NetIF：网络接口是移动节点访问信道的接口。这个接口通过碰撞和无线传输模块来接收其他节点发送到信道上的分组。它将波长、传输功率等信息写入分组头，接收节点的无线传输模块通过分组头中的这些信息来判断分组在到达时的功率是否足够，只有功率大于某临界值时分组才能被正确接收。这个模块的代码位于～ns/mac/phy.{cc，h}、～ns/mac/wireless-phy.{cc，h}。

(6) Antenna：移动节点使用单一增益的全向天线，相关代码位于～ns/mac/antenna.{cc，h}。

(7) 无线信号传输模型：这个模型用来计算每个分组在到达接收节点时的信号强度(功率)。在移动节点的网络接口层有一个接收功率阈值，当接收到的分组的信号强度(功率)小于这个阈值时，这个分组就被标记为 error 并被 MAC 层丢弃掉。NS 中包含了三个无线信号传输模型：Free-space 模型、two-ray-ground 模型和 Shadowing 模型。无线信号传输模型的相关代码位于～ns/mobile/propagation.{cc，h}、tworayground.{cc，h}、shadowing.{cc，h}。

(8) 信道：无线信道的功能是将分组复制给所有连接到本信道上的移动节点(除了分组的源节点)。所有接收到分组的节点需要自己根据无线信号传输模型来判断是否能正确接收到分组。每一个信道对象都会维护一个网络接口对象的列表，列表中包含了所有连到这个信道上的网络接口对象，信道只保存这个列表的头节点指针 ifhead_。通过 ifhead_，信道可以遍历整个列表，这样信道就能实现从一个网络接口对象接收 packet，然后复制 n 份给其他的 n 个网络接口对象。信道的相关代码位于～ns/mac/channel.{cc，h}中。

由于 NS-2 中双模模块的 WLAN 部分是基于 MIP 的，所以下面还要介绍一下 NS-2 中 MIP 的扩展。

NS 中无线模型的 wired-cum-wireless 扩展也为其支持无线 MobileIP 铺好了道路。Sun Microsystem(Charlie Perkins 等)的 MobileIP 模型就是基于 ns 的有线模型(由 Node 和 Link 组成)，因此不需要使用 CMU 的移动模型。在此简要介绍无线 MobileIP 的实现。移动 IP 场景由 HA(home-agents，本地代理)和 FA(foreign-agents，外地代理)，以及在 HA 和 FA 之间运动的 MH(mobile-hosts，移动主机)组成。HA 和 FA 本质上就是基站节点，而 MH 则是移动节点。

MobileIP 扩展的方法和过程参见 mip.{cc，h}、mip-reg.cc、ns-mip.tcl 和 ns-wireless-mip.tcl。HA 和 FA 节点被定义为具有一个 regagent_(注册代理)的 MobileNode/MIPBS

类，能够向移动节点发送 beacon，按照需要建立封装器（encapsulator）和解封器（descpsulator），并回复 MH 的请求。MH 节点被定义为 MobileNode/MIPMH 类，它同样具有一个 regagent_来接收和响应 beacon 并向 HA 或 FA 发送请求。如图 8-10 所示为 Mobile/MIPBS 节点示意图。MobileNode/MIPMH 节点结构和该图相似，只是它没有任何封装器和解封器。

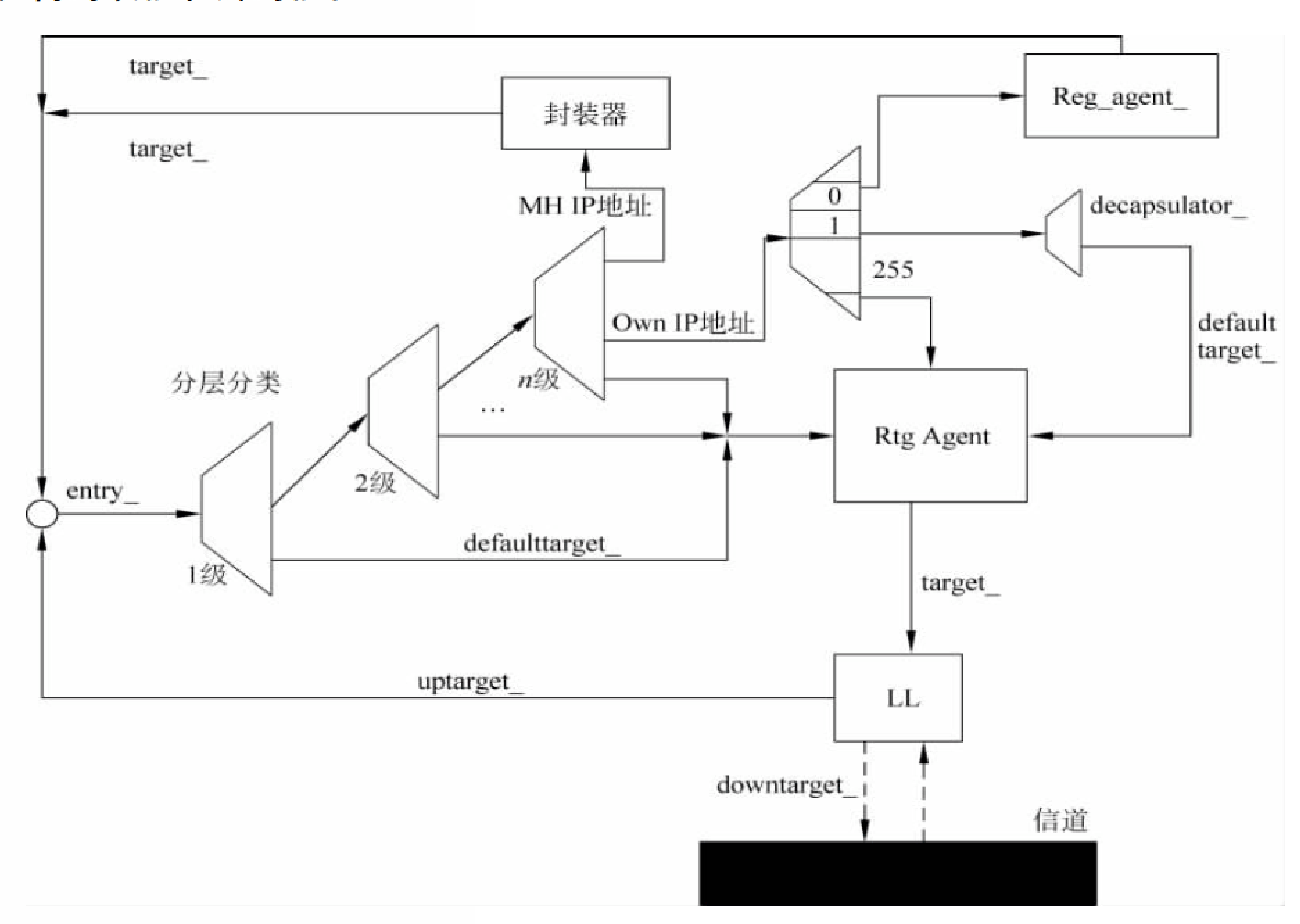


图 8-10　无线 MobileIP 基站节点示意图

2. MIP 的工作原理

MobileNode/MIPBS 节点定期向 MH 广播 beacon 或者 ad(advertisement)消息。

移动节点发来的请求(solicitation)将在基站处产生一个 ad，该 ad 被直接发往请求的 MH。向外发送 beacon 的基站的地址将被 MH 所接收并用作该 MH 的 CoA(care-of-address，转交地址)。因此当 MH 从当前域向外移动到外域时，它的 CoA 也随之改变。基站在接收到一个 MH 的注册请求 reg_request(作为 ad 的回复)时将检查它自己是否为该 MH 的 HA，若不是，它会建立它的解封器并且将该 reg_request 转发给 MH 真正的 HA。

当基站是发出请求的 MH 的 HA 但 CoA 不匹配时，基站就建立一个封装器并把注册请求回复(reg-request-reply)发回给向它转发这个 reg_request 的 FA(以 MH 请求中的 CoA 为地址)。这样所有以该 MH 为目的地址的分组在到达 HA 后，都会经过封装器以隧道的形式先发往 FA(以 CoA 为地址)，而非直接发往目的 MH。封装器将 IP 分组头封装成 IPinIP 分组头。FA 的解封器接收到该分组，将其解封装后发送到目的 MH。

当HA的CoA匹配时，说明该MH已经回到了HA的本地域，于是HA就删除之前(在该MH移动到其他域时)为其建立的封装器，并直接向MH发送回复。

MH在接收不到基站的任何ad时就会向外发送请求。接收到ad时，它就将自己的CoA修改为发送来ad的HA/FA的地址，并且向CoA回复注册请求消息(reg-request)。开始时MH处于自己的HA范围内，并直接从CoA(即此时的HA)处接收所有的发给它的分组。后来随着MH移动出HA的范围而进入到一个FA的外域，MH的CoA也由原来的HA变为FA。然后之前的HA会建立一个封装器以便把所有发往MH的分组转发给FA，而FA则将这些分组解封装后发送给MH。MH想要发送到有线域的数据总是先交由它当前的CoA进行路由转发。

这一部分的工作原理对应的NS代码非常简单，就不在这里具体分析了。

8.3.4 UMTS/WLAN双模网络仿真平台的搭建

未来的移动通信网络将涵盖多种技术、跨多个行业为用户提供高效服务，“融合”将成为未来宽带无线通信发展的必然趋势。由于无线接入技术的多样性，为使用户能够同时使用采用不同接入技术的无线网络，就必须使用户的移动终端同时兼容多个无线接入技术，移动终端将是多模移动终端，以实现全球范围内多个移动网络和无线网络间的无缝漫游，这是用户在不同网络间无缝切换的前提。本小节将重点介绍网络融合的方式，并以WLAN和UMTS为代表介绍蜂窝移动通信系统与无线接入网络的融合方案，同时介绍搭建的UMTS/WLAN双模网络仿真平台。

原有的UMTS模块的致命缺点在于底层的工作机制上，漫游机制的问题在于MAC层对于ARP信令不兼容，主体的改正思路在于interface的添加以及上层Agent的协调。改动重点集中在以下几个方面：

(1) 接口(interface)的协调。一般来说，双模终端的主要工作机制就在于两套接口的协调工作，首先必须分别针对两套不同机制的接口实行分别挂装，其中包括接口的构造以及在形成节点的时候与其他模块之间的连接。两套接口的协调主要体现在Network Agent模块以及ARP模块的机制设计，Network Agent模块是改动的重点，需要通过该模块的控制来实现两套接口的协调，至于ARP方面主要是在于与UMTS各层机制的兼容问题。

(2) 上层Agent的设计。由于双模终端的特性，在上层需要设计两种不同的Agent分别负责WLAN和UMTS，WLAN中原有的模块可以重用，但是UMTS方面是重新设计的，主要是进行一些注册管理以及interface的切换控制(通过控制Network Agent模块)。

下面介绍双模平台搭建过程中的主要步骤。

1. 添加UEAgent、NODEBAgent、GGSNAgent

(1) 在UE上加入上层的管理实体UEAgent，功能为：

- 从下层(RRC)获得所属区域，即所属GGSN、NODEB的信息。
- 进行动态的IP地址的配置。

- 对 UE 的注册以及切换的流程和信令进行管理。

(2) 在 NODEB 上加入上层管理实体 NODEBAgent，功能为：

- 注册等流程的信令管理。
- 隧道的解封装。

(3) 添加 GGSNAgent，功能为：

- 注册等信令管理。
- 实现 GGSN 和 NODEB 之间的隧道的封装。

2. UE 注册流程的修改

原有的注册流程如图 8-11 所示。

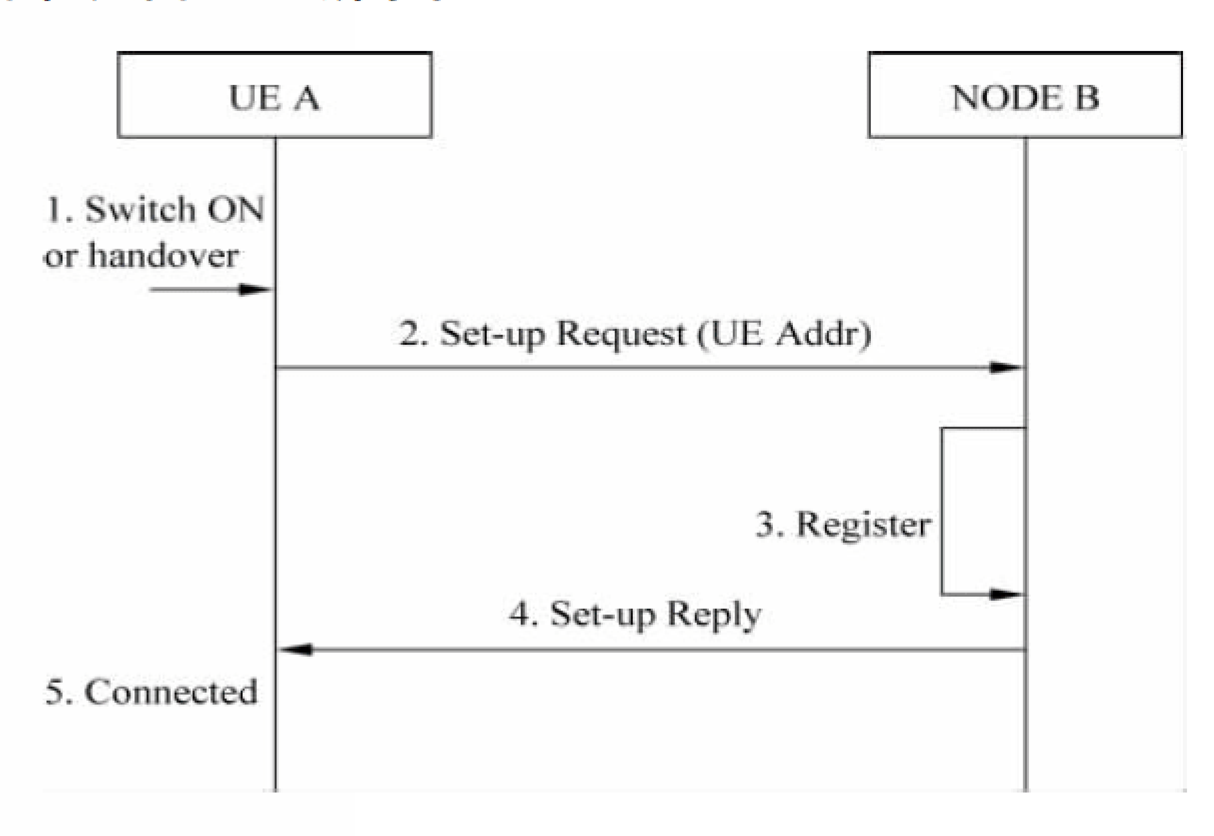


图 8-11　原来的 UMTS 模块的注册流程

原有的注册流程简单，UE 没有地址配置功能，没有 GGSN 实体，没有 GGSN 和 NODEB 之间的隧道。

为了实现隧道机制和 UE 的地址自动配置，对注册流程进行了修改，如图 8-12 所示。

在图 8-12 所示的 4. LL-SETUP-REPLY(GGSN and NODEB IP) 中，信令包含了所在小区的所属 NODEB 和 GGSN 的 IP 地址，UE 获得该信息之后就可以利用 GGSN 的地址前缀自动配置转交地址 GCOA，并利用 NODEB 的 IP 地址自动配置转交地址 LCOA。

在 UE 配置好地址之后，就会给 NODEB 和 GGSN 发送注册请求，分别为 PT_BU_NODEB 和 PT_BU_GGSN。在注册成功之后，GGSN 会将发往 UE 的包通过隧道转给 NODEB，NODEB 解包之后会把包发给 UE。

3. 添加 UE 双 Agent

因为在仿真中终端 UE 需要具备在 UMTS 和 WLAN 中通信的功能，因此就需要两套管理实体，分别管理 WLAN 和 UMTS 中的信令，如图 8-13 所示。

UE 上管理 UMTS 的实体是 UEAgent，加在 Port Classifier dmux_的端口 5。

UE 上管理 WLAN 的实体是 TMAgent，加入 Port Classifier dmux_的端口 0。

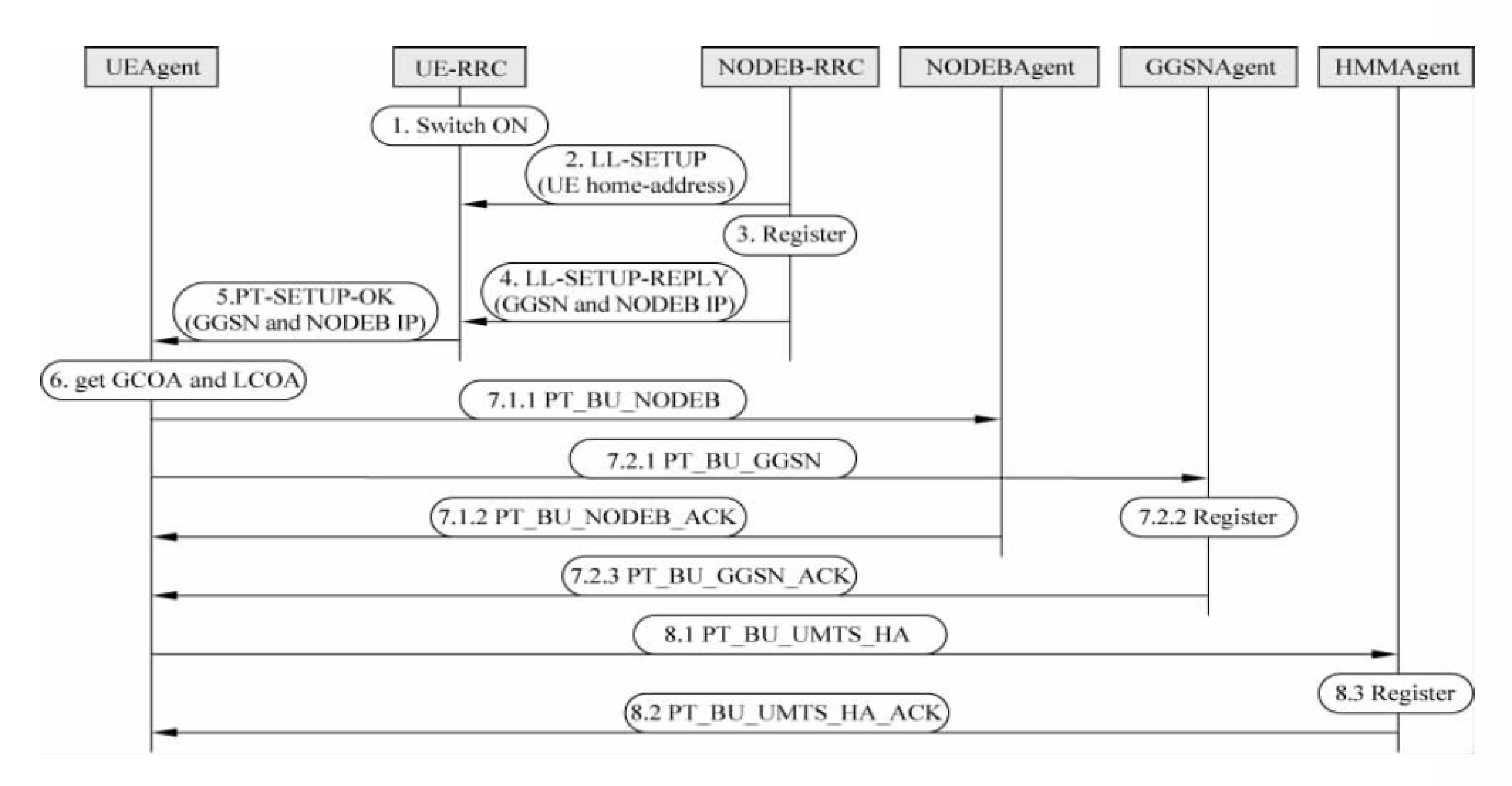


图 8-12　修改后的 UMTS 模块的注册流程

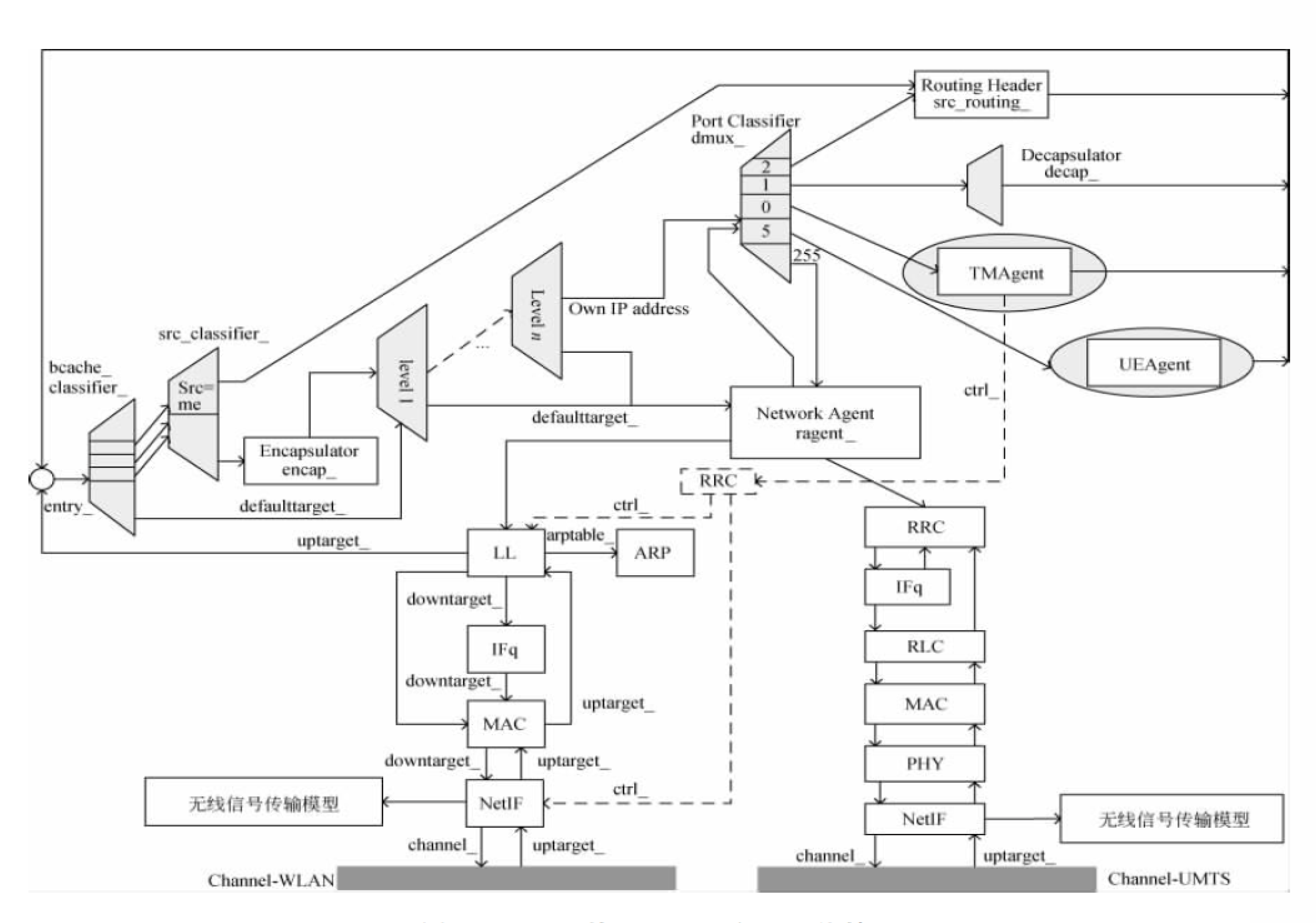


图 8-13　UE 的双 Agent 和双无线接口

4. 添加 UE 双无线接口

因为在仿真中终端 UE 需要具备在 UMTS 和 WLAN 中通信的功能，除了需要两套管理实体来管理 WLAN 和 UMTS 中的信令外，还需要两个无线接口：WLAN 的无线接口和 UMTS 的无线接口。如图 8-13 所示，左下方是 WLAN 的无线接口（包括 LL、IFq、MAC、NetIF），右下方是 UMTS 的无线接口（包括 RRC、IFq、RLC、MAC、PHY、NetIF）。

此外，图 8-13 中，Network Agent 上有指向 WLAN 无线接口的指针和 UMTS 无线接口的指针，当有包到来时，Network Agent 会根据无线接口的启用情况决定将包发向哪个接口。

5. Interface_Selector

由于 UE 上有两个无线接口，需要一个实体根据无线链路的情况决定何时启用哪个接口，这个实体就是 Interface_Selector。

Interface_Selector 中有指向 WLAN 无线接口物理层的指针和指向 UMTS 无线接口物理层的指针，Interface_Selector 定时取得 WLAN 和 UMTS 的信号能量信息，并决定启动哪个无线接口。修改后的机制是：

（1）当 WLAN 接口的信号能量高于 WLAN 的门限时，使用 WLAN 接口。

（2）当 WLAN 接口的信号能量低于 WLAN 的门限时，如果 WLAN 接口信号能量高于 UMTS，则使用 WLAN 接口；如果 WLAN 接口信号能量低于 UMTS，则使用 UMTS 接口。

6. WLAN 和 UMTS 分别用两个信道

因为当使用一个信道时，UMTS 和 WLAN 的无线接口上会收到两个网络（WLAN 和 UMTS）的信令，会发生混乱；而且分开信道可以使仿真更模拟现实，所以决定将 WLAN 和 UMTS 分别用两个信道。

是否使用两个信道的标记位是 ChanIDflag。

（1）初始值为 OFF。

（2）当 WLAN 和 UMTS 要分别使用不同的信道时，在 TCL 脚本中加入“$ns set ChanIDflag ON”。

（3）当使用双无线接口时，需要把 WLAN 和 UMTS 分开信道，所以要把 ChanIDflag 设为 ON。

在 UE 上添加 Timer 来定期向 GGSN 和 NODEB 进行注册和添加 UMTS 注册过程的信令包大小。

7. 路由优化

路由优化就是当 UE 在大域间切换时（AR 和 NODEB 之间），UE 向 CN 发送注册信令，当 CN 收到该消息之后，就把数据包目的地址直接填写为 UE 的转交地址，从而不经过 HA 转包，直接将包发向 UE 所在的外地域。

原来的 WLAN 本身有路由优化机制，只需将 WLAN 的路由优化标志位设置为 1，

就可以启动 WLAN 的路由优化功能。所以修改就集中在 UMTS 的路由优化部分。

修改内容如下：

(1) 加入 UMTS 的路由优化标志位，即 Agent/UEAgent set rt_opti_umts_1。

(2) 路由优化默认为开，关闭时在脚本中加入 Agent/UEAgent set rt_opti__0。

(3) 在 UE 上加入 CN 的 IP 定期更新机制，即当收到数据包时，UE 会记录当前时间和发送该数据包的 CN 的 IP。

(4) 在 UEAgent 中加入给 CN 发送 BU 的部分，即切换过程中在给 HA 发送 BU 的同时，也给 CU 发送 BU。

(5) UE 中加入对 CN 的定期的 BU，用于 CN 定期地绑定更新。

(6) 无论注册还是绑定更新过程，UE 给 CN 发 BU 之前，UE 会检查 CN 的 IP 的最后更新时间，当在一定时间之内，就认为 CN 和 UE 保持链接，从而会给 CN 发送 BU。

(7) 当 UEAgent(UMTS 中 UE 的控制实体)的 CN 的 IP 更新时，同时更新 TMAgent(WLAN 中 UE 的控制实体)的 CN 的 IP 信息。

做了以上修改之后，发现在 UE 从 AR 切换到 NODEB，当 HA 和 UE 的距离相对 CN 和 UE 的距离较远时，CN 比 HA 先收到 BU，从而开始直接给 UE 发送数据包，但此时 CN 到 HA 的链路上已经有一部分先前发送给 HA，需要 HA 转发给 UE 的包，这部分数据包在 HA 收到 BU 之后，会被改变转包地址(HA 收到 BU 之前转发给 AR，收到 BU 后转发给 NODEB)，因此，NODEB 会在切换过程中给 UE 发送两个链路来的包，一个是 CN 直接到 UE(最内层 IP 包的目的地址为 UE 的转交地址)，一个是 CN 到 HA 再到 UE(最内层 IP 包的目的地址为 UE 的家乡地址)。因为原来在 NODEB 上的转包队列是基于最内层 IP 包的目的地址，从而在 NODEB 上会为 UE 分配两个队列，这两个队列的包在 RLC 层分片的序号都会由 0 开始。当数据分片到达 UE 的 RLC 层时，由于流 ID 相同，UE 的 RLC 层会将相同流 ID 的序号重叠的数据片扔掉。

为了避免数据丢失，又对 NODEB 做了修改，让 NODEB 把所有发向 UE 的包都排入一个队列，解决了上面的丢包问题。

8.4 仿真实例

本节主要介绍 WLAN 和 UMTS 融合平台上仿真实例和性能分析。

8.4.1 基于移动 IP 的垂直切换仿真

首先对基于移动 IP 的 UMTS 与 WLAN 之间的垂直切换进行仿真分析。

1. 仿真场景

如图 8-14 所示是基本移动 IP 的仿真场景拓扑图，该场景包括节点 cn1_、ha1_、router1_、ggsn1_、nodeb1_、ar1_和移动节点 ue1_。链路的设置为：

(1) cn1_与 router1_之间的链路带宽为 100Mb/s，时延为 5ms、15ms、25ms、50ms 4 种情况。

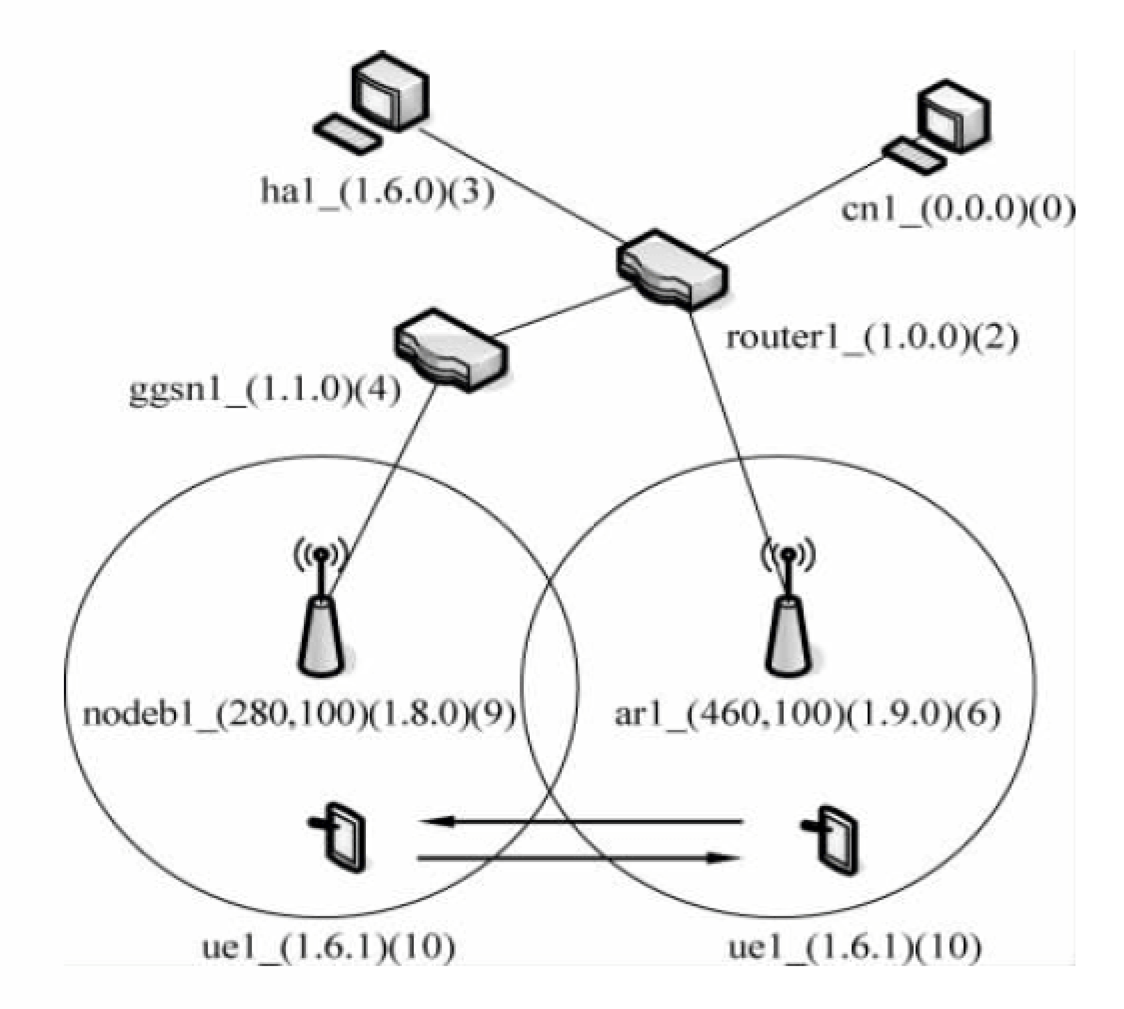


图 8-14　添加路由优化模块的 UMTS 与 WLAN 之间的切换场景

（2）router1_与 ha1_之间的链路带宽为 100Mb/s，时延为 5ms、15ms、25ms 三种情况。

（3）router1_与 ggsn1_之间的链路带宽为 100Mb/s，时延为 2ms。

（4）router1_与 ar1_之间的链路带宽为 100Mb/s，时延为 2ms。

（5）ggsn1_到 nodeb1_之间的链路时延为 5ms、15ms 两种情况。

2．切换流程分析

各流程的具体描述如下（与图 8-15 相对应）：

（1）Ipv6_sol——mn 发向接入路由器 ar1 的路由器请求消息。

（2）Ipv6_rads——ar1 发出的路由器通告消息。

（3）Mipv6_bu——mn 发向家乡代理的绑定更新消息。

（4）Mipv6_back——家乡代理发向 mn 的绑定确认消息。

（5）Cbr——通信对端 cn 发向 mn 的 cbr 包。

（6）setup_ok——nodeb 通知 mn 链路建立成功。

（7）Bu_nodeb——mn 发向 nodeb 的绑定更新消息。

（8）Bu_nodeb_ack——nodeb 发向 mn 的绑定确认消息。

（9）Bu_ggsn——mn 发向 ggsn 的绑定更新消息。

（10）Bu_ggsn_ack——ggsn 发向 mn 的绑定确认消息。

（11）Bu_umts_ha——mn 发向家乡代理 ha1 的绑定更新消息。

（12）Bu_umts_ha_ack——家乡代理 ha1 发向 mn 的绑定确认消息。

（13）Mipv6_bu——mn 发向 cn 的绑定更新消息。

（14）Cbr——通信对端 cn 发向 mn 的 cbr 包。

（15）Ipv6_sol——mn 发向接入路由器 ar1 的路由器请求消息。

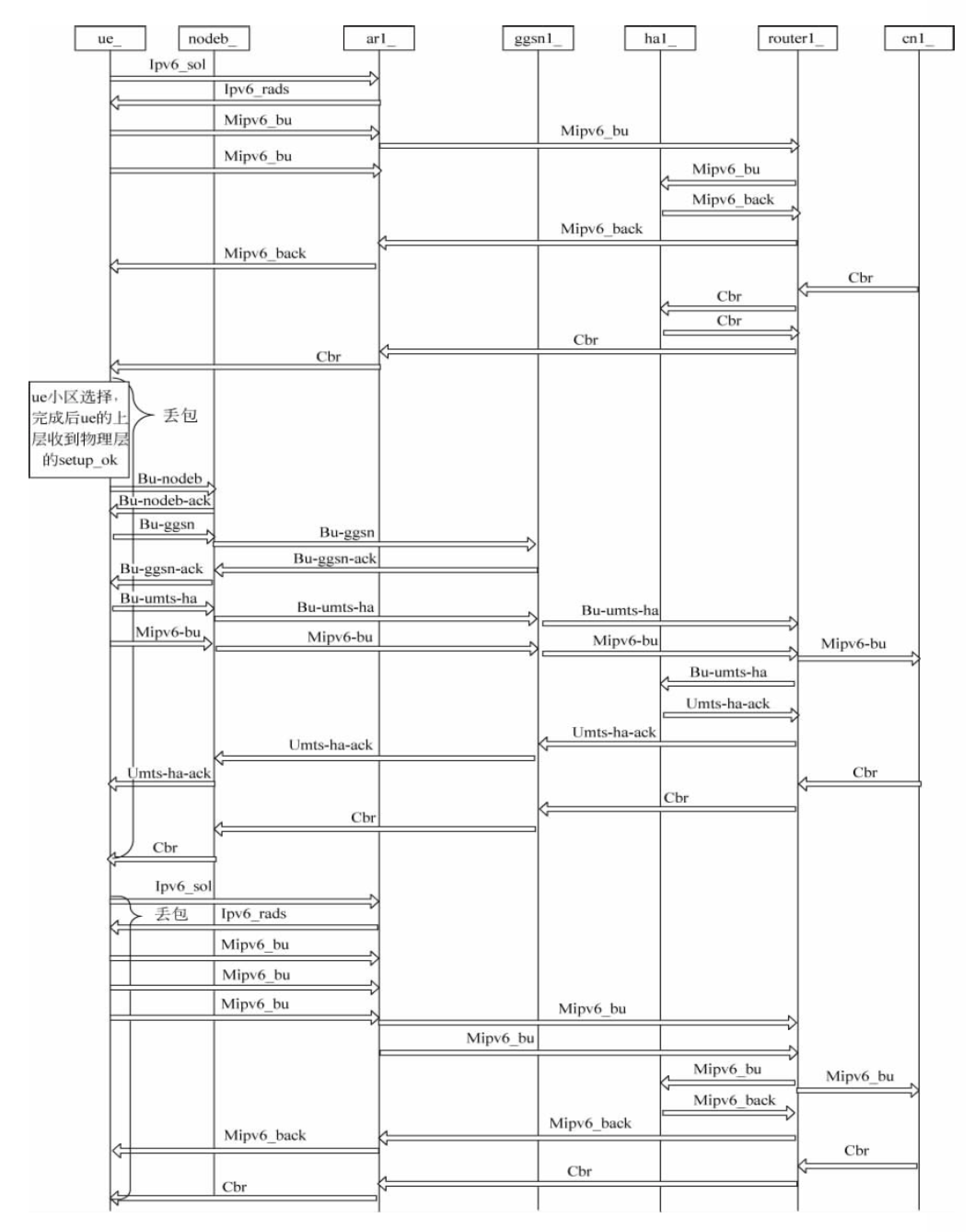


图 8-15　WLAN 切换到 UMTS 再切换回 WLAN 的流程图

（16）Ipv6_rads——ar1 发出的路由器通告消息。

（17）Mipv6_bu——mn 发向家乡代理的绑定更新消息。

（18）Mipv6_bu——mn 发向 cn 的绑定更新消息。

（19）Mipv6_back——家乡代理发向 mn 的绑定确认消息。

（20）Cbr——通信对端 cn 发向 mn 的 cbr 包。

本场景分析的是 ue1_从 WLAN 切换到 UMTS 再切换回 WLAN 的过程。

具体来说，第一阶段是 ue1_从 ar1_切换到 nodeb1_，在此过程中 ue1_完成注册、切换，接收 cn1_发送的路由优化后的 cbr 包，在 nodeb1_进行解包。过程中发现有丢包。

第二阶段是 ue1_从 nodeb1_切换到 ar1_，在此过程中 ue1_完成注册、切换，接收 cn1_发送的 cbr 包，在 ue1_进行解包。过程中发现有丢包。

3. 仿真结果和分析

仿真主要分析切换过程中隧道建立的时间和注册完成的时间以及切换过程中的丢包个数。

1）ue1_从 ar1_切换到 nodeb1_

在 ue1_从 ar1_切换到 nodeb1_的时延由下面几个部分组成，如图 8-16 所示。

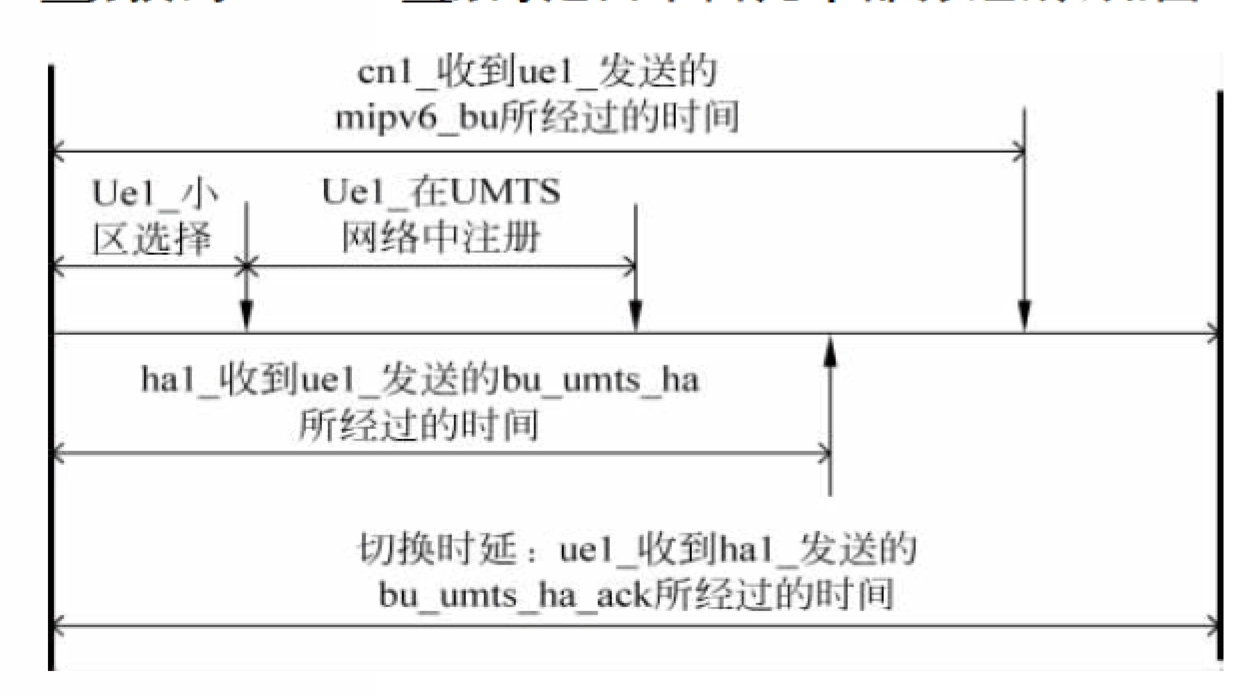


图 8-16　ue1_从 ar1_切换到 nodeb1_的时延分析

（1）ue1_小区选择，是 ue1_的 UMTS 无线接口进行的对 nodeb 的选择。

（2）ue1_在 UMTS 网络中注册——从 ue1_发送 bu_ggsn，到 ue1_收到 ggsn1_返回的 bu_ggsn_ack 所经过的时间。

（3）cn1_建立路由优化的时间——从 ue1_开始小区选择，到 cn1_收到 ue1_发送的 mipv6_bu 所经过的时间。

（4）ha1_隧道建立的时间——从 ue1_开始小区选择，到 ha1_收到 ue1_发送的 bu_umts_ha 所经过的时间。

（5）切换时延——从 ue1_开始小区选择，到 ue1_收到 ha1_发送的 bu_umts_ha_ack 所经过的时间。

表 8-2～表 8-5 中，τ1 代表 cn1_与 router1_之间的链路时延，单位为 ms，取值有 5、15、25、50；τ2 代表 router1_与 ha1_之间的链路时延，单位为 ms，取值有 5、15、25；τ3 代表 ggsn1_到 nodeb1_之间的链路时延，单位为 ms，取值有 5、15。

表 8-2 ue1_从 ar1_切换到 nodeb1_,ue1_在 UMTS 网络中注册的时间(ms)

(即从 ue1_发送 bu_ggsn,到 ue1_收到 ggsn1_返回的 bu_ggsn_ack 所经过的时间)

$\tau 3$ \ $\tau 1/\tau 2$	5/5	5/15	5/25	15/5	15/15	15/25
5	17.342	17.342	18.009	18.009	17.342	17.342
15	37.352	37.352	38.019	38.019	37.352	37.352
$\tau 3$ \ $\tau 1/\tau 2$	25/5	25/15	25/25	50/5	50/15	50/25
5	17.342	17.342	18.009	18.009	18.009	17.342
15	37.352	37.352	38.019	38.019	38.019	37.352

从表 8-2 中可以看出,ue1_**从** ar1_**切换到** nodeb1_,**注册的时间只取决于** ggsn1_**和** nodeb1_**之间的链路时延**($\tau 3$)。**当** $\tau 3$ **为** 5ms **时**,ue1_**在** UMTS **网络中注册的时间约为** 18ms;**当时延为** 15ms **时**,ue1_**在** UMTS **网络中注册的时间约为** 38ms。

表 8-3 ue1_从 ar1_切换到 nodeb1_,cn1_建立路由优化的时间(ms)

(即从 ue1_开始小区选择,到 cn1_收到 ue1_发送的 mipv6_bu 所经过的时间)

$\tau 3$ \ $\tau 1/\tau 2$	5/5	5/15	5/25	15/5	15/15	15/25
5	48.242	48.242	48.825	58.825	58.242	58.242
15	78.242	78.242	78.825	88.825	88.242	88.242
$\tau 3$ \ $\tau 1/\tau 2$	25/5	25/15	25/25	50/5	50/15	50/25
5	68.242	68.242	68.825	93.825	93.825	93.242
15	98.242	98.242	98.825	123.825	123.825	123.242

从表 8-3 中可以看出,ue1_**从** ar1_**切换到** nodeb1_,cn1_**建立路由优化的时间取决于** cn1_**与** router1_**之间的链路时延**($\tau 1$)**和** ggsn1_**和** nodeb1_**之间的链路时延**($\tau 3$)。

表 8-4 ue1_从 ar1_切换到 nodeb1_,ha1_隧道建立的时间(ms)

(即从 ue1_开始小区选择,到 ha1_收到 ue1_发送的 bu_umts_ha 所经过的时间)

$\tau 3$ \ $\tau 1/\tau 2$	5/5	5/15	5/25	15/5	15/15	15/25
5	48.386	58.386	68.969	48.969	58.386	68.386
15	78.413	88.413	98.969	78.969	88.413	98.143
$\tau 3$ \ $\tau 1/\tau 2$	25/5	25/15	25/25	50/5	50/15	50/25
5	48.386	58.386	68.969	48.969	58.969	68.386
15	78.413	88.413	98.969	78.969	88.969	98.413

从表 8-4 中可以看出,ue1_**从** ar1_**切换到** nodeb1_,ha1_**隧道建立的时间取决于** router1_**与** ha1_**之间的链路时延**($\tau 2$)**和** ggsn1_**和** nodeb1_**之间的链路时延**($\tau 3$)。

表 8-5 ue1_从 ar1_切换到 nodeb1_,切换时延(ms)

(即从 ue1_开始小区选择,到 ue1_收到 ha1_发送的 bu_umts_ha_ack 所经过的时间)

τ1/τ2 \ τ3	5/5	5/15	5/25	15/5	15/15	15/25
5	71.352	92.029	112.039	72.019	91.362	112.039
15	111.372	131.382	152.059	112.039	131.382	151.392
τ1/τ2 \ τ3	**25/5**	**25/15**	**25/25**	**50/5**	**50/15**	**50/25**
5	71.352	91.362	112.039	72.019	92.029	111.372
15	111.372	131.382	152.059	112.039	131.165	151.392

从表 8-5 中可以看出,ue1_从 ar1_切换到 nodeb1_,切换时延取决于 router1_与 ha1_之间的链路时延(τ2)和 ggsn1_和 nodeb1_之间的链路时延(τ3)。

如图 8-17 所示是切换时延图。从中可以明显看到切换时延随 τ2 和 τ3 的变化。

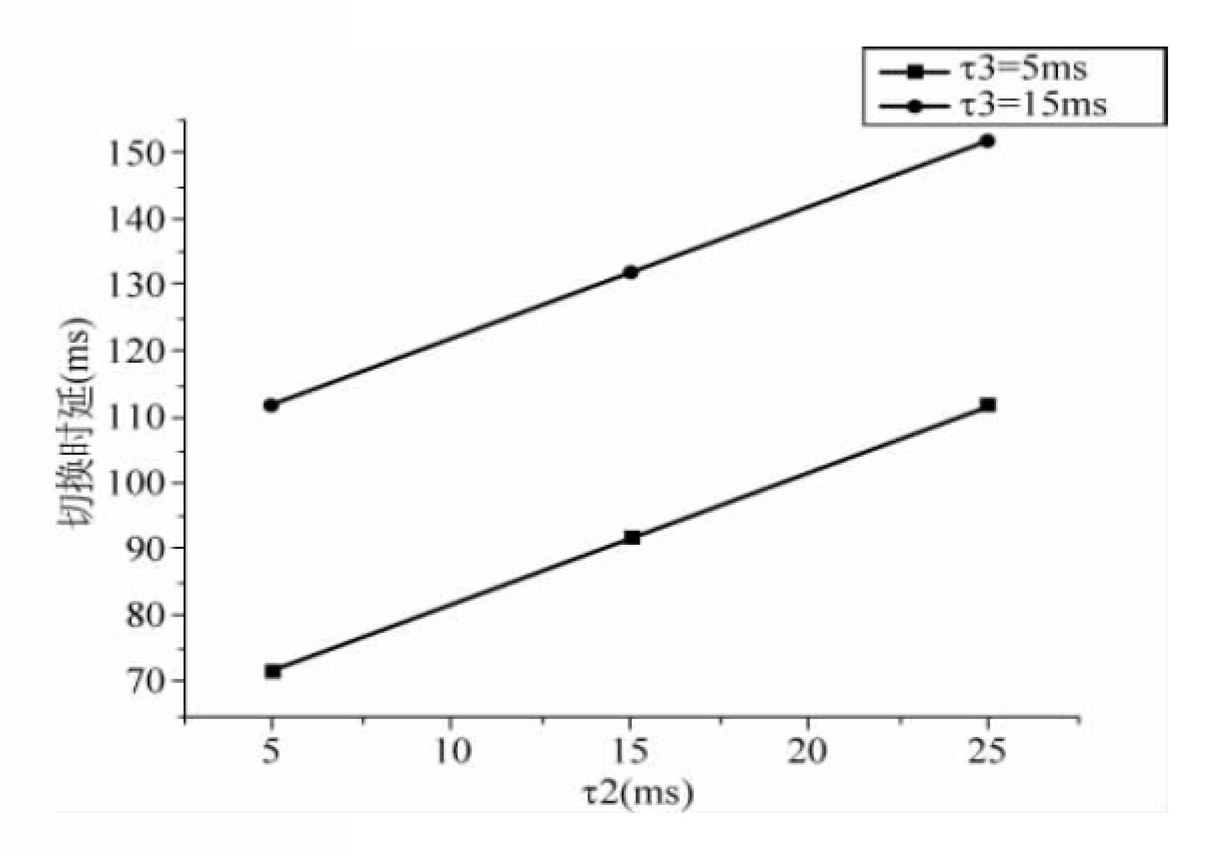


图 8-17 ue1_从 ar1_切换到 nodeb1_切换时延

2) ue1_从 nodeb1_切换到 ar1_

ue1_从 nodeb1_切换到 ar1_的时延由下面几个部分组成,如图 8-18 所示。

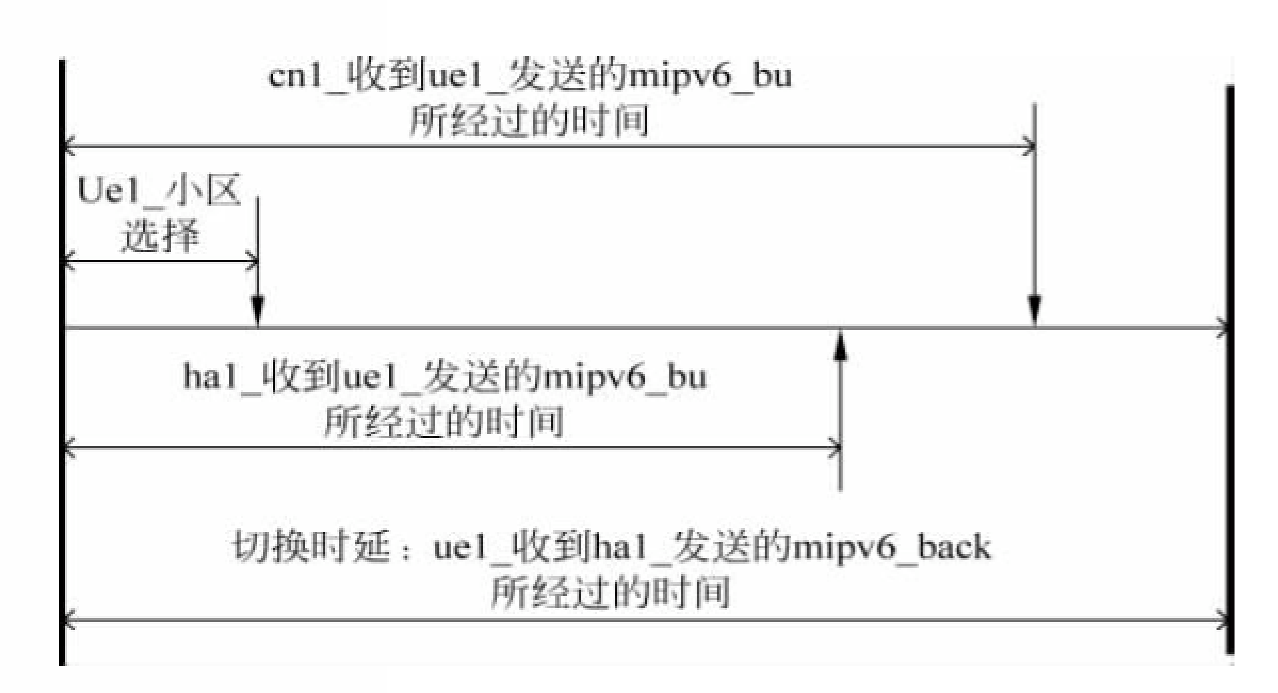


图 8-18 ue1_从 nodeb1_切换到 ar1_的时延分析

(1) ue1_小区选择，是 ue1_的 WLAN 无线接口进行的对 nodeb 的选择。

(2) cn1_建立路由优化的时间——从 ue1_发送 ipv6_sol，到 cn1_收到 ue1_发送的 mipv6_bu 所经过的时间。

(3) ha1_隧道建立的时间——从 ue1_发送 ipv6_sol，到 ha1_收到 ue1_发送的 mipv6_bu 所经过的时间。

(4) 切换时延——从 ue1_发送 ipv6_sol，到 ue1_收到 ha1_发送的 mipv6_back 所经过的时间。

表 8-6～表 8-8 中，τ1 代表 cn1_与 router1_之间的链路时延，单位为 ms，取值有 5、15、25、50；τ2 代表 router1_与 ha1_之间的链路时延，单位为 ms，取值有 5、15、25；τ3 代表 ggsn1_到 nodeb1_之间的链路时延，单位为 ms，取值有 5、15。

表 8-6　ue1_从 nodeb1_切换到 ar1_，cn1_建立路由优化的时间(ms)
(即从 ue1_发送 ipv6_sol，到 cn1_收到 ue1_发送的 mipv6_bu 所经过的时间)

τ1/τ2 \ τ3	5/5	5/15	5/25	15/5	15/15	15/25
5	8.771	8.201	8.971	18.611	18.752	19.776
15	8.131	8.252	8.291	18.511	18.771	18.331
τ1/τ2 \ τ3	25/5	25/15	25/25	50/5	50/15	50/25
5	28.202	28.612	28.351	53.771	53.852	52.962
15	28.592	28.491	28.712	53.682	54.291	53.112

从表 8-6 中可以看出，ue1_从 nodeb1_切换到 ar1_，cn1_建立路由优化的时间只取决于 cn1_与 router1_之间的链路时延(τ1)。当 τ1 为 5ms 时，cn1_建立路由优化的时间约为 8.5ms；当 τ1 为 15ms 时，cn1_建立路由优化的时间约为 19ms；当 τ1 为 25ms 时，cn1_建立路由优化的时间约为 28.4ms；当 τ1 为 50ms 时，cn1_建立路由优化的时间约为 53ms。

表 8-7　ue1_从 nodeb1_切换到 ar1_，ha1_隧道建立的时间(ms)
(即从 ue1_发送 ipv6_sol，到 ha1_收到 ue1_发送的 mipv6_bu 所经过的时间)

τ1/τ2 \ τ3	5/5	5/15	5/25	15/5	15/15	15/25
5	7.54	17.41	27.839	7.559	17.82	28.352
15	7.42	17.62	27.94	7.6	17.54	27.42
τ1/τ2 \ τ3	25/5	25/15	25/25	50/5	50/15	50/25
5	7.41	17.48	27.5	7.939	17.92	27.41
15	7.83	17.7	27.92	7.41	17.959	27.62

从表 8-7 中可以看出，ue1_从 nodeb1_切换到 ar1_，ha1_隧道建立的时间只取决于 router1_与 ha1_之间的链路时延(τ2)。当 τ2 为 5ms 时，ha1_隧道建立的时间约为 7.6ms；当 τ2 为 15ms 时，ha1_隧道建立的时间约为 17.6ms；当 τ2 为 25ms 时，ha1_隧道建立的时间约为 27.6ms。

表 8-8　ue1_从 nodeb1_切换到 ar1_,注册完成的时间(ms)
(即从 ue1_发送 ipv6_sol,到 ue1_收到 ha1_发送的 mipv6_back 所经过的时间)

τ1/τ2 　τ3	5/5	5/15	5/25	15/5	15/15	15/25
5	24.689	44.967	65.06	24.708	44.969	65.501
15	24.569	44.769	65.594	24.749	44.689	64.925

τ1/τ2 　τ3	25/5	25/15	25/25	50/5	50/15	50/25
5	24.559	44.629	64.648	25.088	45.069	64.559
15	25.009	44.849	65.069	24.559	45.108	64.769

从表 8-8 中可以看出,ue1_从 nodeb1_切换到 ar1_,切换时延只会取决于 router1_与 ha1_之间的链路时延(τ2)。当 τ2 为 5ms 时,切换时延约为 24.7ms;当 τ2 为 15ms 时,切换时延约为 44.8ms;当 τ2 为 25ms 时,切换时延约为 65ms。

如图 8-19 所示是切换时延图。

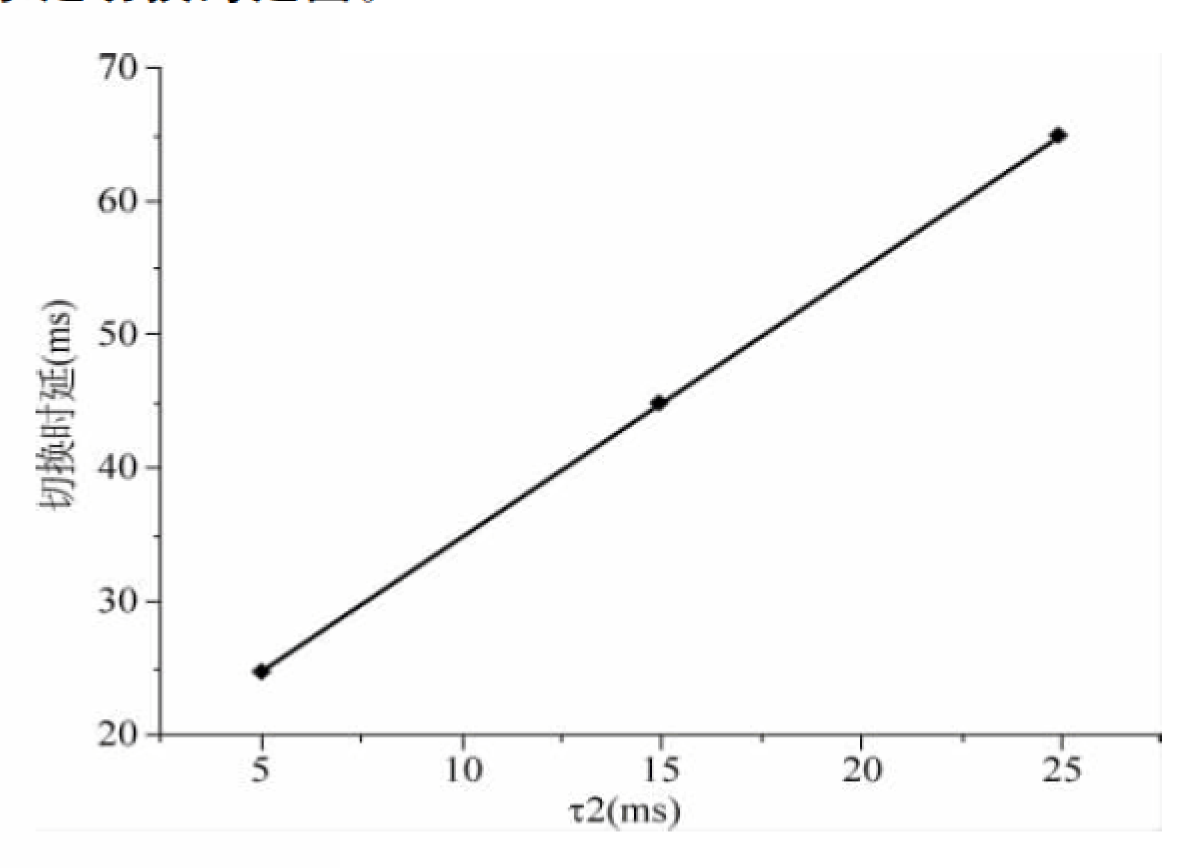


图 8-19　ue1_从 nodeb1_切换到 ar1_切换时延

如图 8-20 所示是 ue1_在两个方向切换时切换时延的比较图。从中可以看出,ue1_从 ar1_切换到 nodeb1_的切换时延要大,而且会受到 τ3 的影响。因为 ue1_从 ar1_切换到 nodeb1_时,要先在 UMTS 网络中注册,所以切换时延会比从 nodeb1_切换到 ar1_要大。又因为 ue 在 UMTS 网络中注册的时间是与 τ3 有关的,当 τ3 增大时,时间就会增大,ue1_从 ar1_切换到 nodeb1_的切换时延也会随之增大。

8.4.2　多接入选择模块

在支持 UMTS/WLAN 双模的网络仿真平台的基础上,添加了多模终端的动态接入网络选择判决模块,进一步进行了通用多接入选择建模的研究仿真。终端所添加的接入选择模块分为多接入选择的数学建模模块和多接入选择的算法实现模块,如图 8-21 所示。

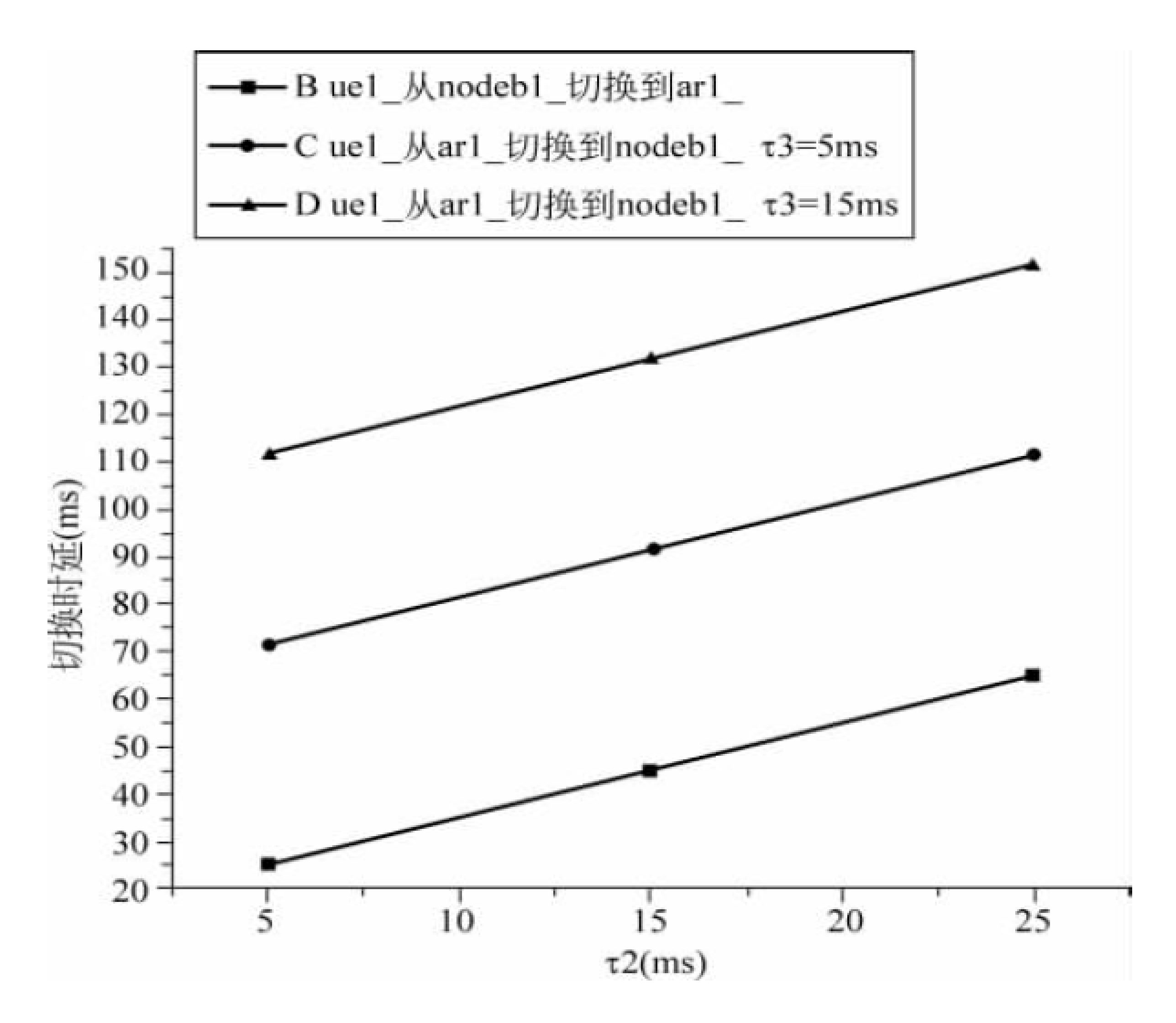


图 8-20 ue1_在两个方向上切换时延的比较

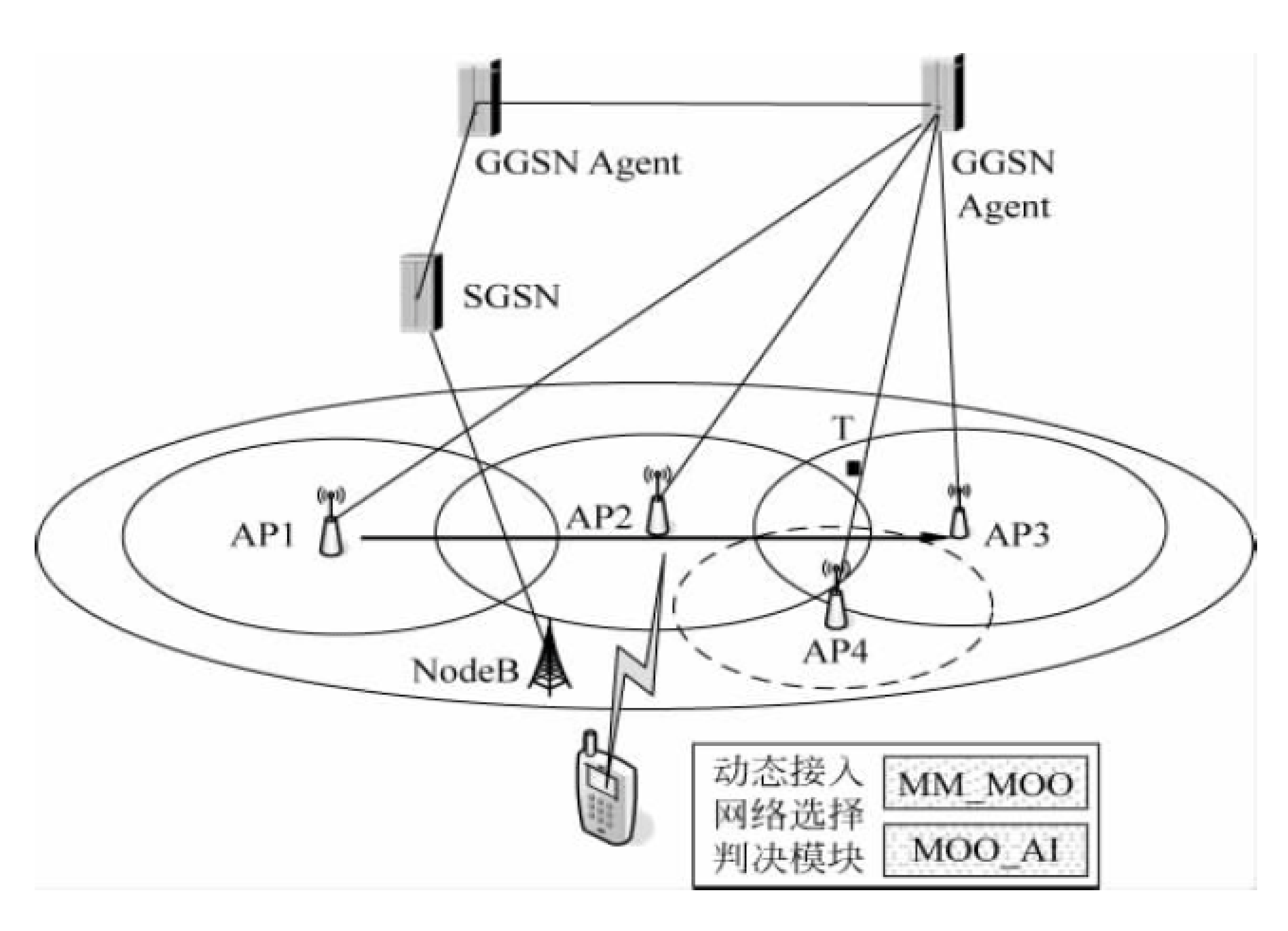


图 8-21 多接入选择模块在仿真平台上的实现

(1) 多接入选择的 MM_MOO(mathematical modelling for multi-objective optimization)模块：位于多模终端上，该模块负责监测网络当前的连接状态；收集可用网络的网络参数，包括网络带宽、时延、可靠性等；对网络参数进行归一化处理，从而实现多接入选择问题到多目标优化问题的建模。

(2) 多接入选择的 MOO_AI(multi-objective optimization_algorithm implementation)模块：位于多模终端上，实现群体排序、层次分析法(analytic hierarchy process，AHP)、Hooke-Jeeves 算法。该模块对网络的参数进行数据处理，得到各个网络的评价值，从而选择应该接入的目标网络。

根据相应仿真文档的数据构建仿真过程的节点拓扑图如图 8-22 所示。其中，AP1、AP2、AP3、AP4 为 4 个 WLAN 的接入点，提供一定范围内的高速率服务，接入同一个 GGSN1。AP 通过发送路由广播使得进入其覆盖范围的 MN 检测到 AP 的存在。NodeB 为 UMTS 基站，通过 SGSN 接入 GGSN2。GGSN1 与 GGSN2 实际通过核心网相连，在仿真拓扑中认为直接相连。

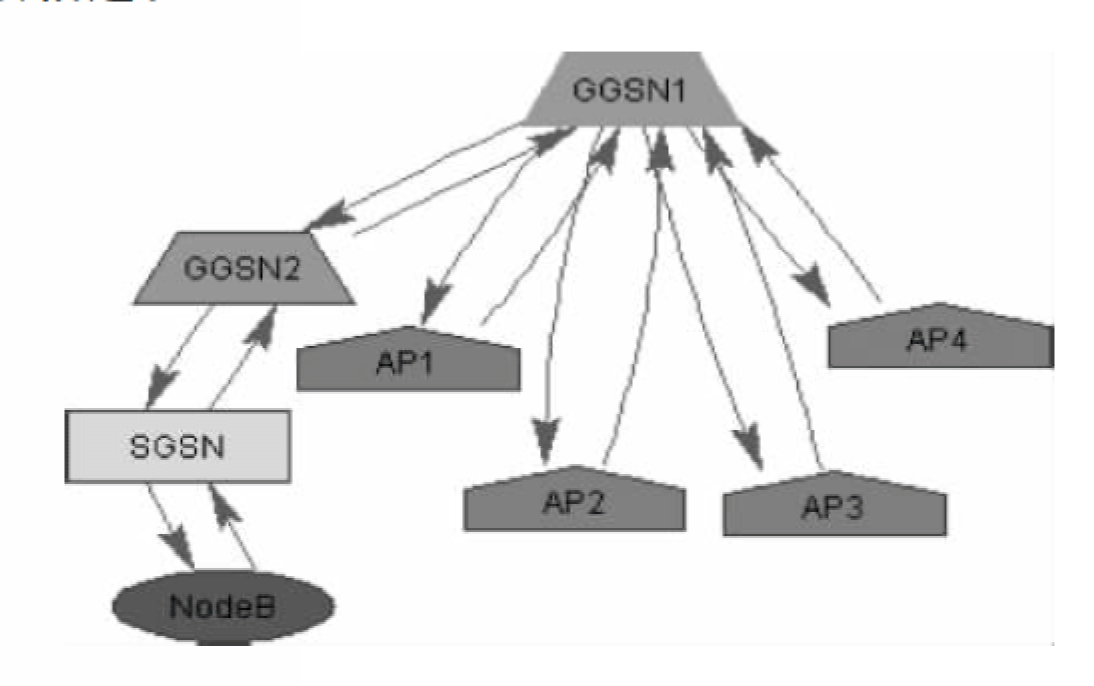


图 8-22　仿真过程的节点拓扑图

根据异构网络的特点，构建图 8-23 所示的仿真场景，其中 UMTS 坐标为(500,0)，覆盖半径为 500m；AP1 坐标为(250,0)，AP2 坐标为(450,0)，AP3 坐标为(700,0)，AP4 坐标为(650,－100)，覆盖半径都为 200m。多模终端通过 MM_MOO 模块监测网络状态，搜集网络参数，并对参数进行归一化处理，当检测到多个可用网络时，触发启用 MOO_AI 模块，通过群体排序层次分析法以及 Hooke-Jeeves 算法，将多目标问题转化为单值问题，其仿真结果如图 8-23 所示。

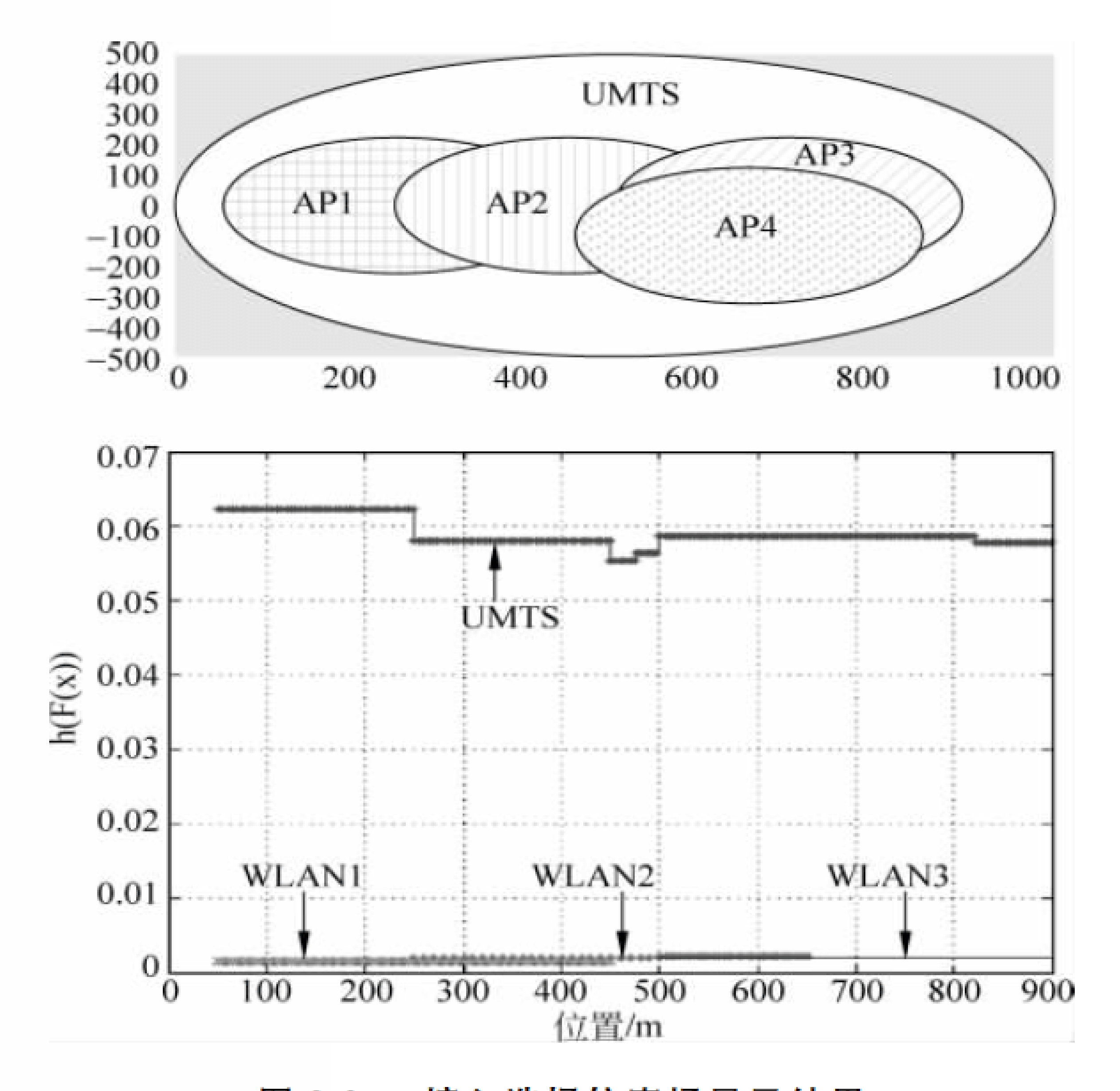


图 8-23　接入选择仿真场景及结果

从图 8-23 中可以看出，在 UMTS 网络与 WLAN 网络重叠覆盖范围内，WLAN 网络普遍能够提供更好的性能。在[50m，900m]的重叠区间中，对 WLAN 网络进行计算，得到仿真结果如图 8-24 所示。

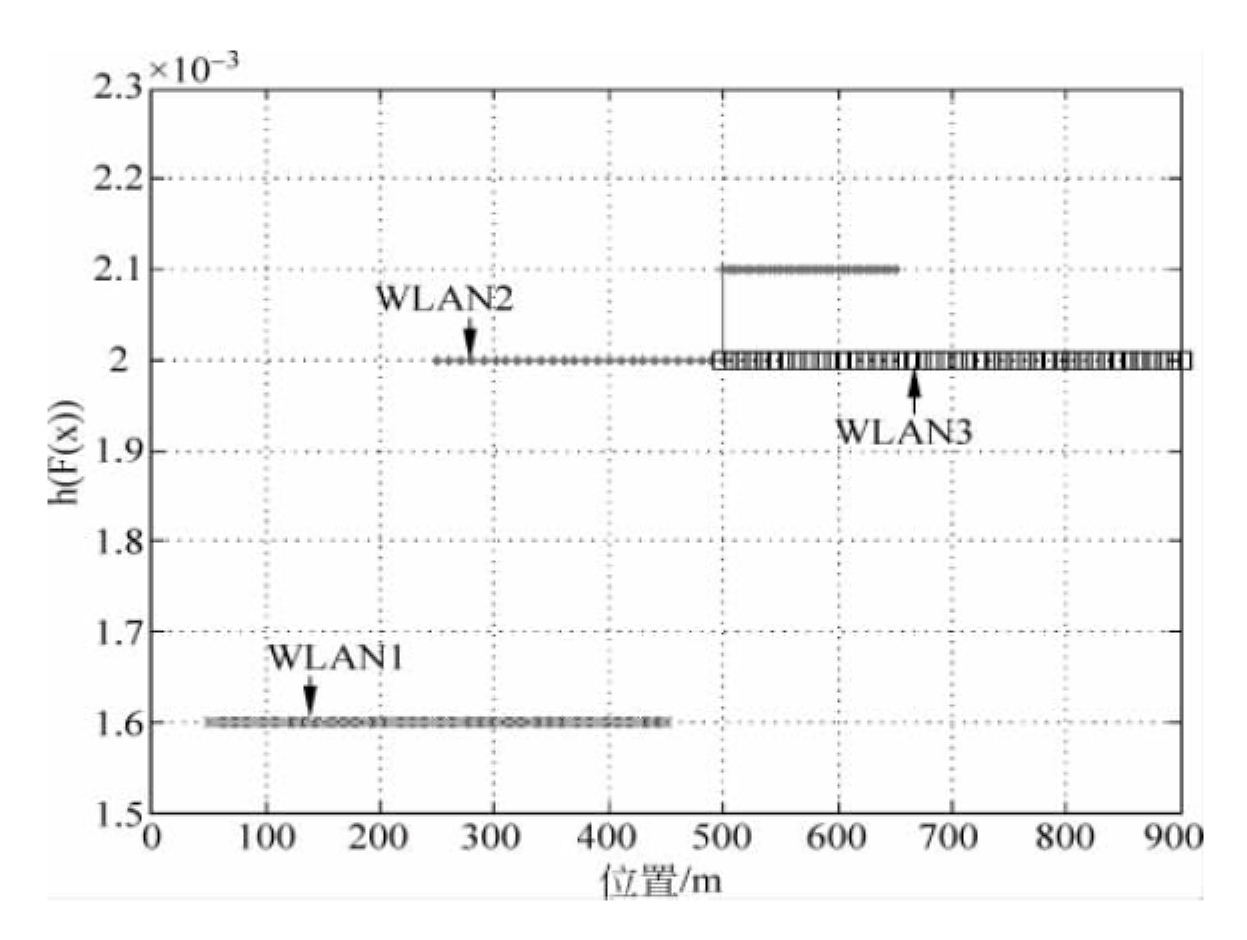


图 8-24　只考虑 WLAN 网络的接入选择仿真结果

从图 8-24 中可以看出，在[50m，450m]范围内，选择 WLAN1 网络；在[450m，500m]范围内，选择 WLAN2 网络；在[500m，900m]范围内，选择 WLAN3 网络。仿真结果如图 8-25 所示。

随着移动节点 MN 的运动，MN 会根据网络的不同而选择最优的网络。在仿真平台的表现上，则将随着 MN 的不断运动，改变节点 MN 的颜色，同时将其所选择网络变成与其自身相同的颜色。

8.4.3　协同无线资源管理模块

基于 UMTS-WLAN 互通 NS-2 仿真平台，添加了协同无线资源管理模块(Co-RRM Model)。考虑到 UMTS 和 WLAN 紧耦合环境下的协同资源管理架构，把 Co-RRM 模块划分为三个功能实体，将其分布在仿真网络平台上的不同代理上，如图 8-26 所示。

(1) 协同管理实体(Co-RRM)部分：考虑到 RMMA 的 MAP 属性，Co-RRM 功能设置在区域移动性代理(RMMA)中，负责 UMTS 网络以及 WLAN 网络中的移动性管理，有效地收集它所管辖区域内的所有子网信息，对不同类型网络的无线资源进行协调分配。

(2) 本地特定的管理实体(S-RRM)：S-RRM 功能设置在 MMA 上，包括 UMTS 网络中的基站(NodeB)和 WLAN 网络中的接入路由器(AR)，负责管理该区域内无线资源。S-RRM 执行各自网络类型的无线资源管理功能，为终端执行水平多接入网络选择，同时能够通过信息报告功能向 Co-RRM 汇报管辖区域子网的负载状况等网络信息。

(3) 终端管理实体(T-RRM)：T-RRM 功能设置在终端移动性代理(TMA)中，主要完成协同无线资源管理请求的触发以及请求消息的封装。TMA 工作在双无线接口状态

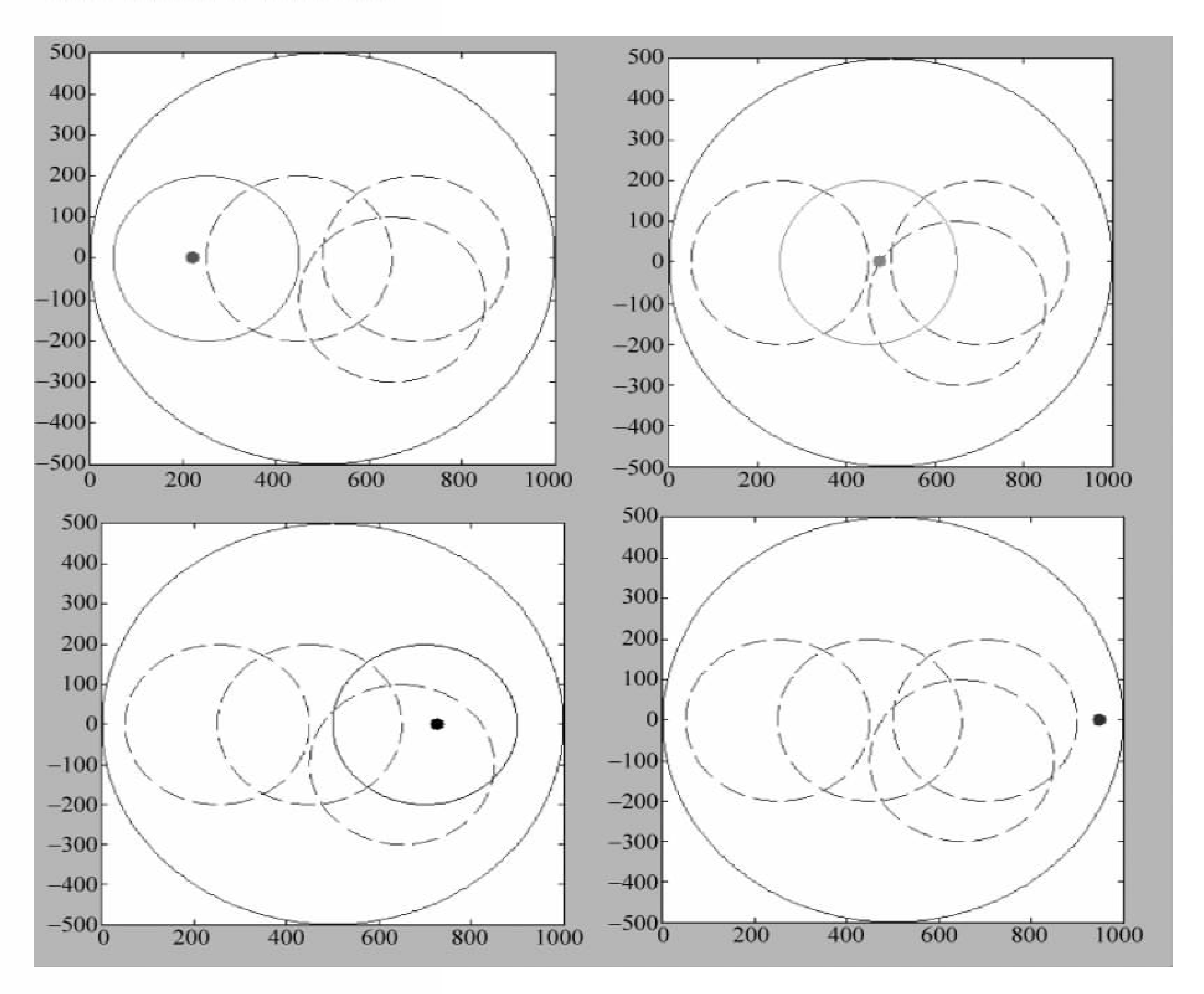


图 8-25　仿真结果的呈现

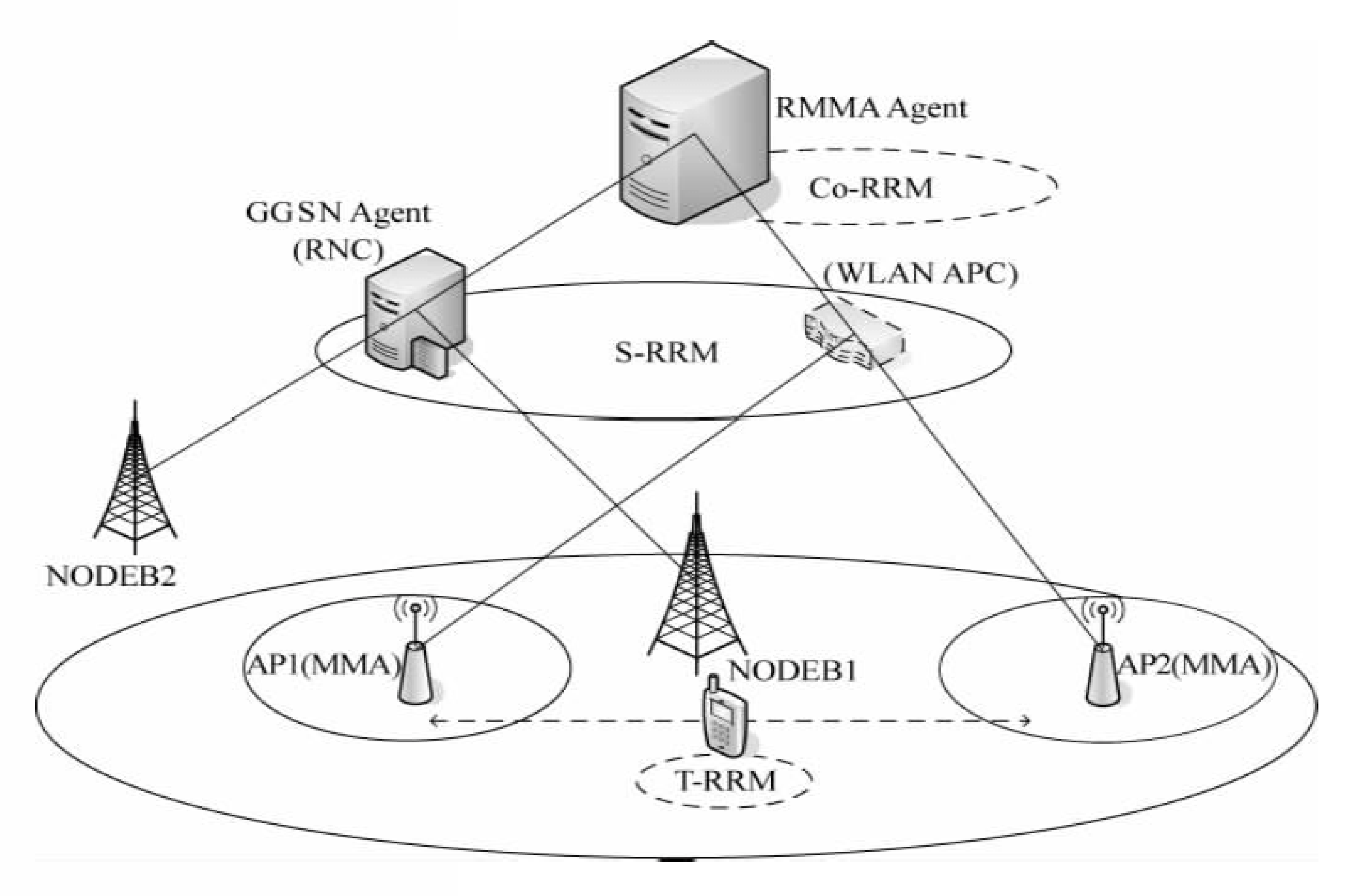


图 8-26　协同无线资源模块在仿真平台上功能实体分布

下，当它监测到多种网络的小区广播而判断进入多网络重叠覆盖区域，或者有新的业务触发寻求接入时，会触发并封装资源请求（R-REQ）消息。R-REQ 消息会通过当前连接的 NodeB 或 MMA，或者当前信号最强的候选子网络向上传递给 S-RRM 以及 Co-RRM。

R-REQ 消息的包格式如图 8-27 所示。

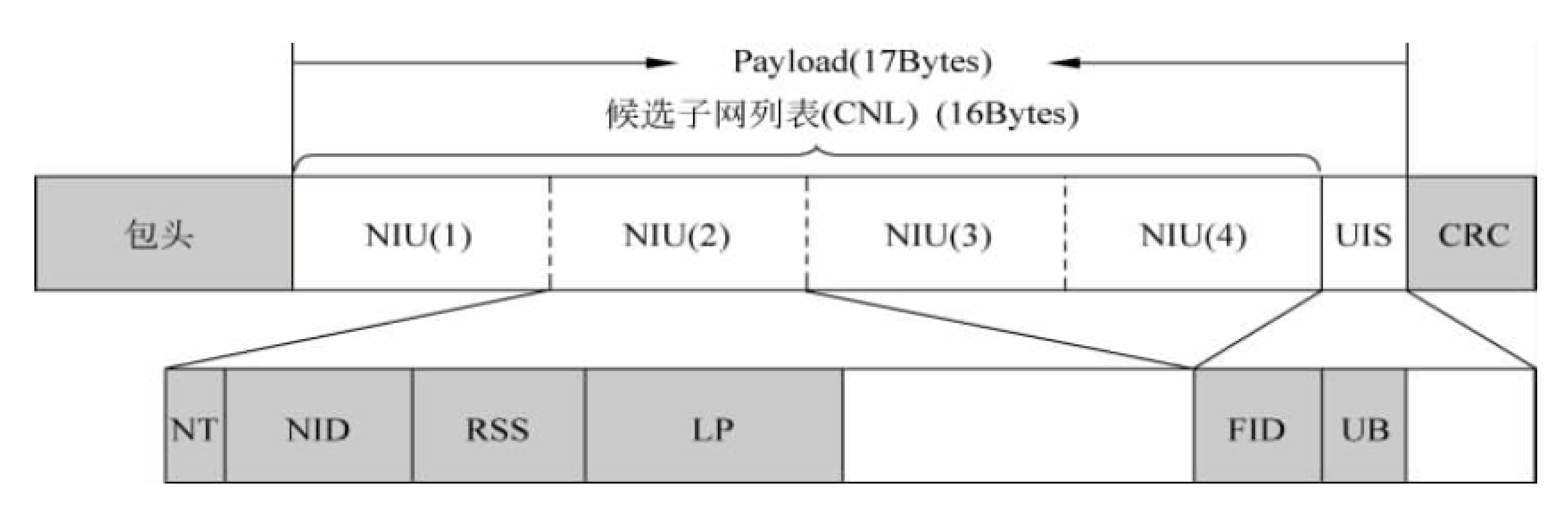


图 8-27 协同无线资源管理请求消息格式

R-REQ 消息的包格式主要分为 4 个部分：

（1）消息包头（header）——包括消息标识、消息编号，还有一些路由信息，包括目标 RRM 实体的 IP 地址、下一跳地址等。

（2）候选子网列表（candidate sub-network list，CNL）——大小设定为 16 个字节，每个子网信息单元（sub-network information unit，NIU）包含 4 个字节，一共可以容纳 4 个候选子网信息单元。填充时按照终端测量网络列表的顺序，也就是按照接收信号强度（RSS）大小依次递减排列。每个子网信息单元 NIU（共 32 比特）的构成如下：候选子网类型（sub-network type，NT），采用 2 个比特来标识网络类型（如 UMTS 0，WLAN 1，WiMAX 2）；候选子网标识，即子网 ID 号（sub-network ID，NID），采用 6 个比特统一编码当前网络中的小区数，一共可以标识 128 个小区；终端测量的网络参数，目前定义为接收信号强度（received signal strength，RSS），因为每种网络的信号强度绝对值不具有可比性，进行归一化处理后采用 4 比特编码，分为 16 个信号强度等级；子网负载情况，因为每种网络的负载绝对值不具有可比性，采用各网络的负载占总容量的百分比即 LP（load percentage）表示负载信息，采用 7 比特编码，当 S-RRM 向 Co-RRM 发送 R-REQ 时添加此项，因为终端无法测量网络容量，所以 T-RRM 发出的 R-REQ 此项为全零；另外，13 个比特目前空闲，留给以后填充其他网络信息，如网络价格、带宽等参数。

（3）用户信息序列（user information sequence，UIS）——大小为 1 个字节，包括当前业务流标识，或者发起新业务请求类型（service flow ID，FID），假设 8 种业务，采用 3 比特编码；用户的网络选择偏好 UB（user bias），为 4 个等级，采用 2 比特表示；另外 3 个比特备用。

（4）冗余校验部分 CRC——对传输过程中可能产生的误码进行循环冗余校验，确保消息传递正确。

协同无线资源管理方法的具体实施流程如图 8-28 所示，包括以下步骤：

（1）当移动终端产生新业务，或进入多网络重叠覆盖的区域并且收到的各网络信号强度大于该网络信号阈值（大于阈值能够使用该网络进行通信，小于阈值无法连接到该网络）持续一段时间，终端无线资源管理（T-RRM）功能实体就开始测量和收集各无线网

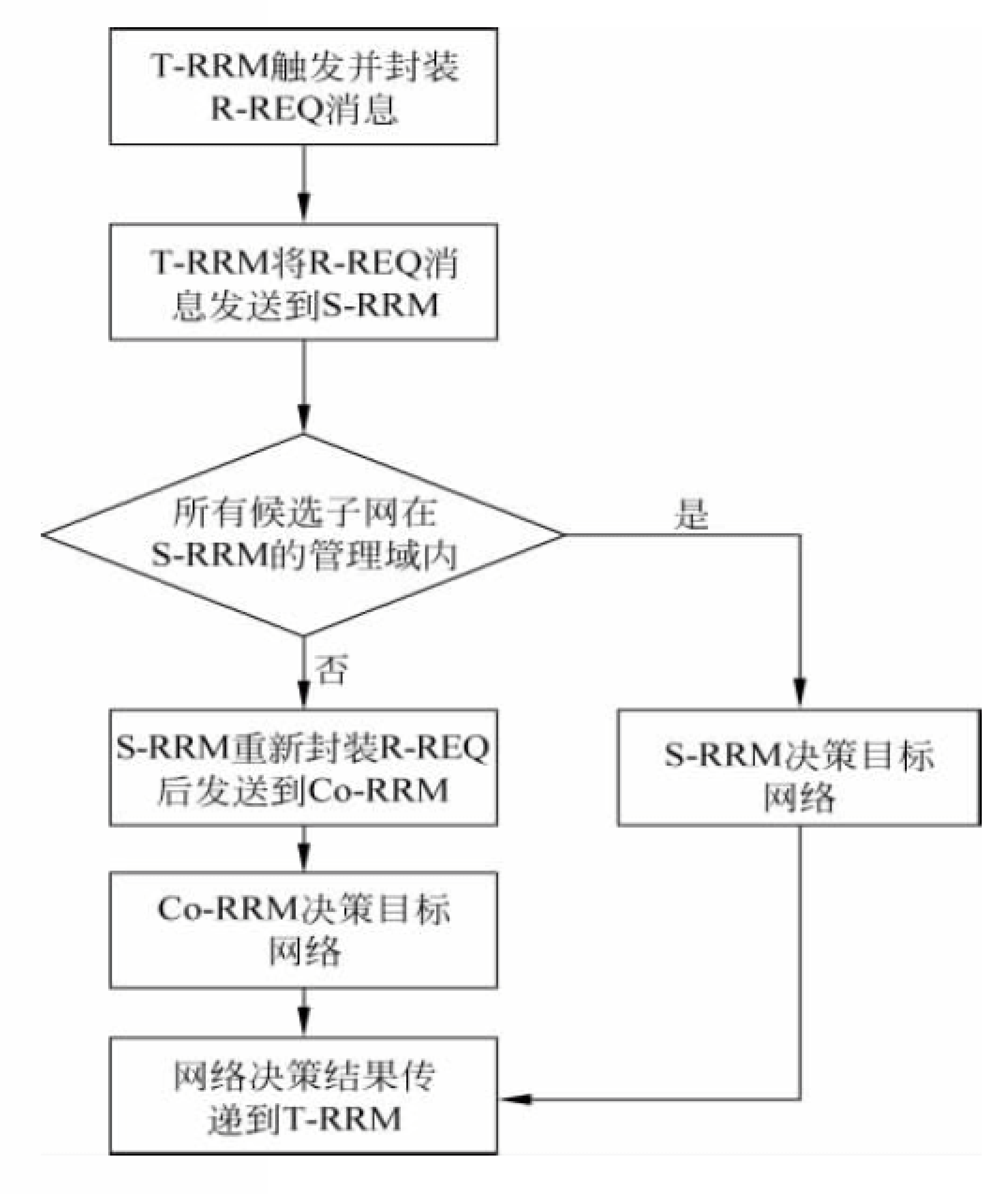


图 8-28　协同无线资源管理的实施流程

络的参数和业务特性及用户偏好，之后触发资源请求（R-REQ）消息，该请求消息包含一个候选网络列表，该列表主要包括可用网络标识、终端测得的网络参数（例如，信号强度、误码率等）以及基本的用户信息（例如，发起请求的业务类型、用户的网络选择偏好和用户标识等）。

（2）T-RRM 把该 R-REQ 消息发送给信号质量最佳或者当前有连接的本地特定无线资源管理（S-RRM）功能实体。

（3）S-RRM 检查候选网络列表中所有候选子网是否都在其管理范围内，如果在其管理域内则进行步骤 4，否则进行步骤 5。

（4）S-RRM 基于资源分配算法判决目标网络并进行步骤 7。

（5）添加本地网络状态信息（例如，容量、小区负载、覆盖区域、带宽、价格等）后重新封装 R-REQ 消息，通过信息报告功能发送给协同无线资源管理（Co-RRM）功能实体；Co-RRM 收到所述 R-REQ 消息后，向其管理域内各候选网络的 S-RRM 发送信息请求，各候选网络的 S-RRM 通过信息报告功能向 Co-RRM 报告本地网络状态信息，如果某些候选网络不在其管理域内，Co-RRM 向其他 Co-RRM 发送信息请求，相应的 Co-RRM 会报告其管理域内各候选网络的 S-RRM 的网络状态信息。

（6）Co-RRM 基于资源分配策略判决目标网络并进行步骤 7。

（7）网络决策结果通过 3 层协议传输给 T-RRM，T-RRM 执行接入请求或垂直切换。

仿真过程中，首先根据仿真文档的信息，读出数据构建仿真场景。本节采用了如图 8-29 所示的仿真场景，其中 MN1、MN2、MN3、MN4 为 4 个多模终端。AP1、AP2 为

两个接入点，其会周期地发出路由广播，在 MN 收到路由广播后，会向 MAP 上报其可用 AP 以及其参数。MAP 则会根据一些特定算法，考虑到 MN 的 QoS 需求以及 AP 的负载状况等因素为 MN 选择合适的 AP。

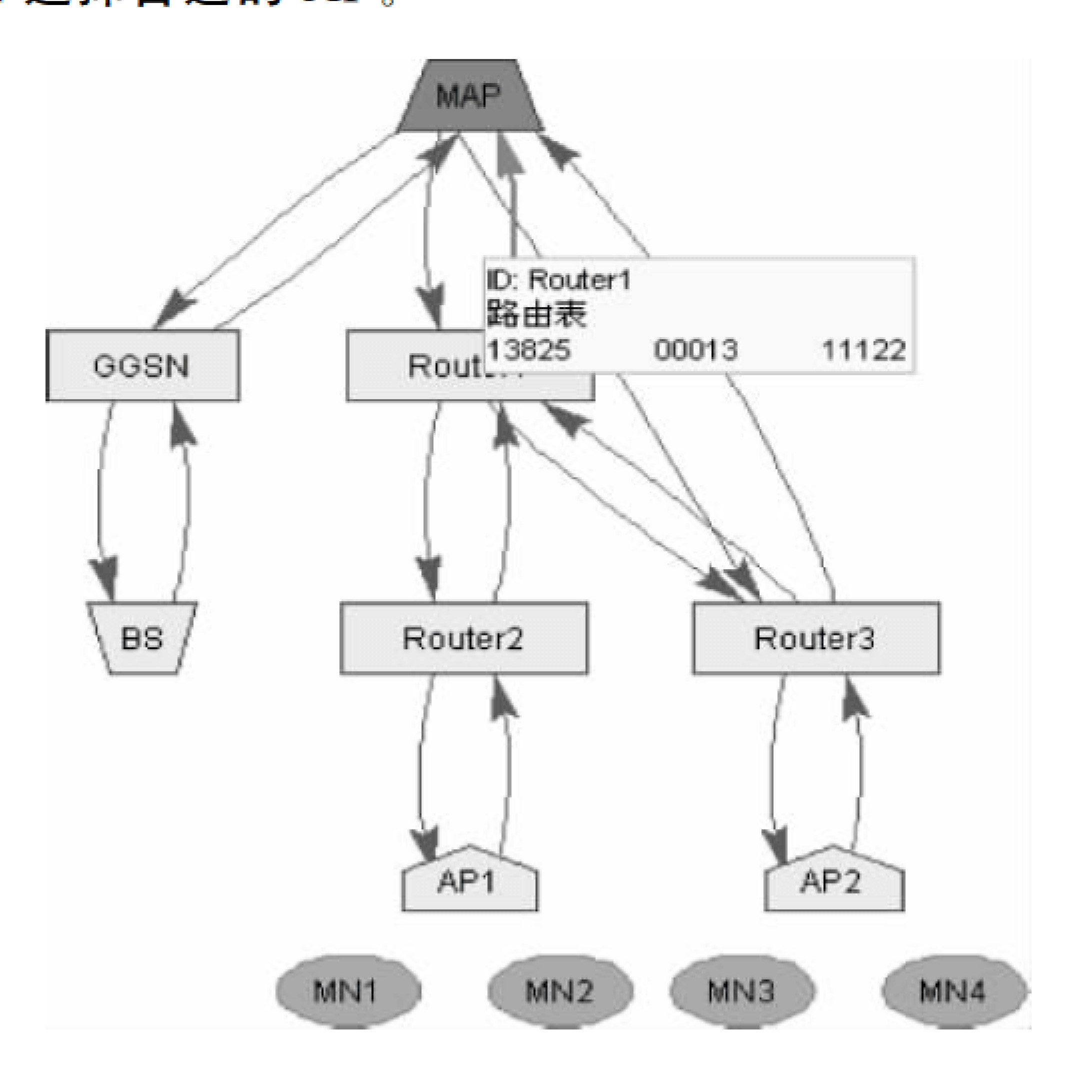


图 8-29 协同无线资源管理仿真场景

在仿真前，网络首先进行路由过程，在各个网络节点上生成路由表。这样，网络中的信息就可以根据路由表进行转发。如图 8-29 所示，Router1 到 MAP 的箭头表示有路由更新发送。在鼠标划过网络单元时，将显示网络节点当前的路由表。

在网络仿真的同时，也有事件列表会被打印在命令窗口。如图 8-30 中阴影部分所示，send_net @ 7 表示在 AP1 的网络层生成一个路由广播。send_mac @ 7 to 0 表示 AP1 的 MAC 层中生成了一个广播事件。在仿真平台中，整个仿真过程中的所有事件都将在控制台上打印出来。这样，通过对事件的分析，就可以了解仿真场景的整个实现过程以及对仿真结果进行验证。

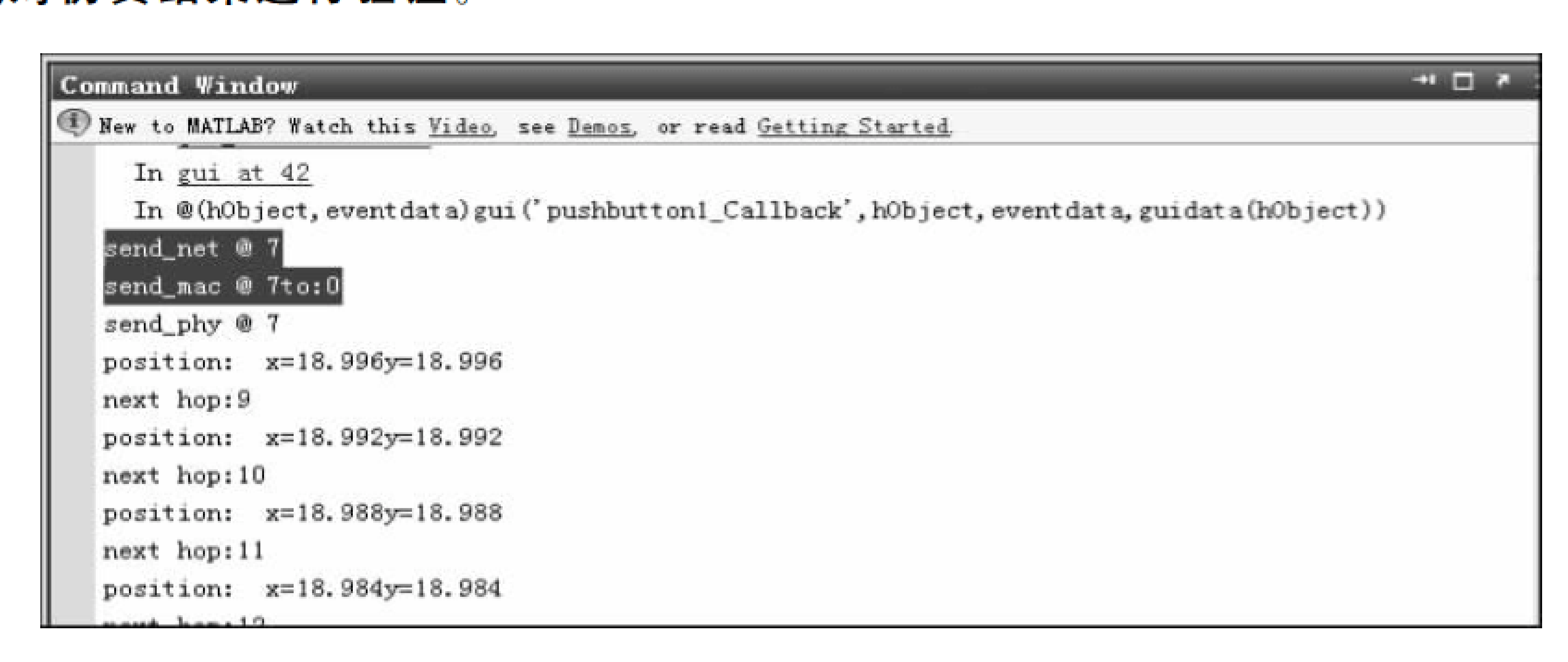


图 8-30 无线资源管理仿真中路由广播发送事件

最后的仿真结果如图 8-31 所示。MN 的参数信息发送到 MAP 后，MAP 将根据所得到的信息，按照事先设计好的算法，对 MN 的 AP 进行指派。在这个仿真中，由于 MN3 的运动，MAP 将其附着的 AP 由 AP2 改变为 AP1，如图 8-31 中阴影部分所示。其中，node 10 为 MN3，node 8 和 node 7 为 AP2 和 AP1。

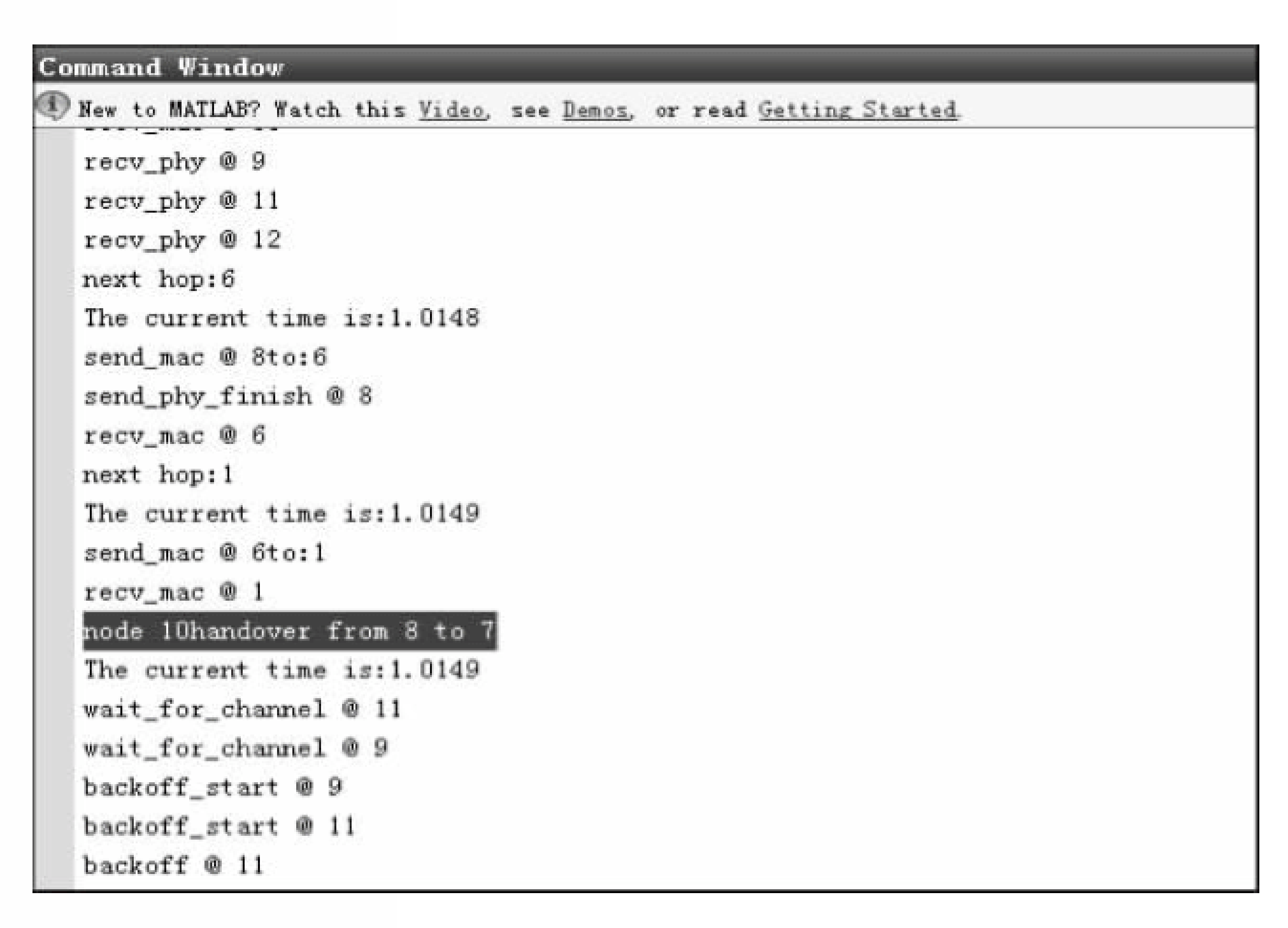


图 8-31　无线资源管理中的切换触发事件

8.4.4　统一的 QoS 保障和管理仿真

基于底层模型及课题组所构建的异构无线全 IP 网络可扩展仿真平台，通过在相关实体上修改函数和添加相关脚本，实现联合分级接入控制流程和算法，根据信号强度、网络负载和用户偏好等参数，在提高业务接纳概率的同时，保障业务接入最优网络，有效地支持并验证了所提出的接入 QoS 方案，即分级联合接入控制框架及其 HJCAC 算法。

原有的基本仿真平台中缺少 HJCAC 的相关功能，因此需要修改相关代码，添加新的消息、信令流程、功能函数及判决算法，构成 HJCAC 功能模块。所构建的 HJCAC 功能模块包括终端 MN、LCAC 和 JCAC 三个功能实体，相关功能和算法分别在仿真网络平台的不同代理上实现，如图 8-32 所示。

（1）MN 模块：终端 MN 实体负责接纳控制管理请求的触发、消息报文封装以及响应，辅助网络进行初始的接入选择。移动节点 MN 处于 UMTS 网络中的 UE 代理部分或处于 WLAN 网络中的 TMA 代理部分。Network Agent 实体负责上层和无线接口之间的交流，并决定包的转发方向；在双模终端的无线接口选择模块加入 HJCAC 的触发以及报文构造函数，当 MN 终端进入子网重叠覆盖区域且有业务生成时，定时获取接收到的小区信号强度，能够进行初步的小区选择，并触发接纳控制管理请求消息 SER-REQ，并根据判决反馈 ADA-REQ 消息，启动相应的接入过程。同时设置触发标识符 sol_flag=1，启动定时器。若在规定时间内没有收到判决反馈，则触发识示符归零，终端将再次发送管理请求。

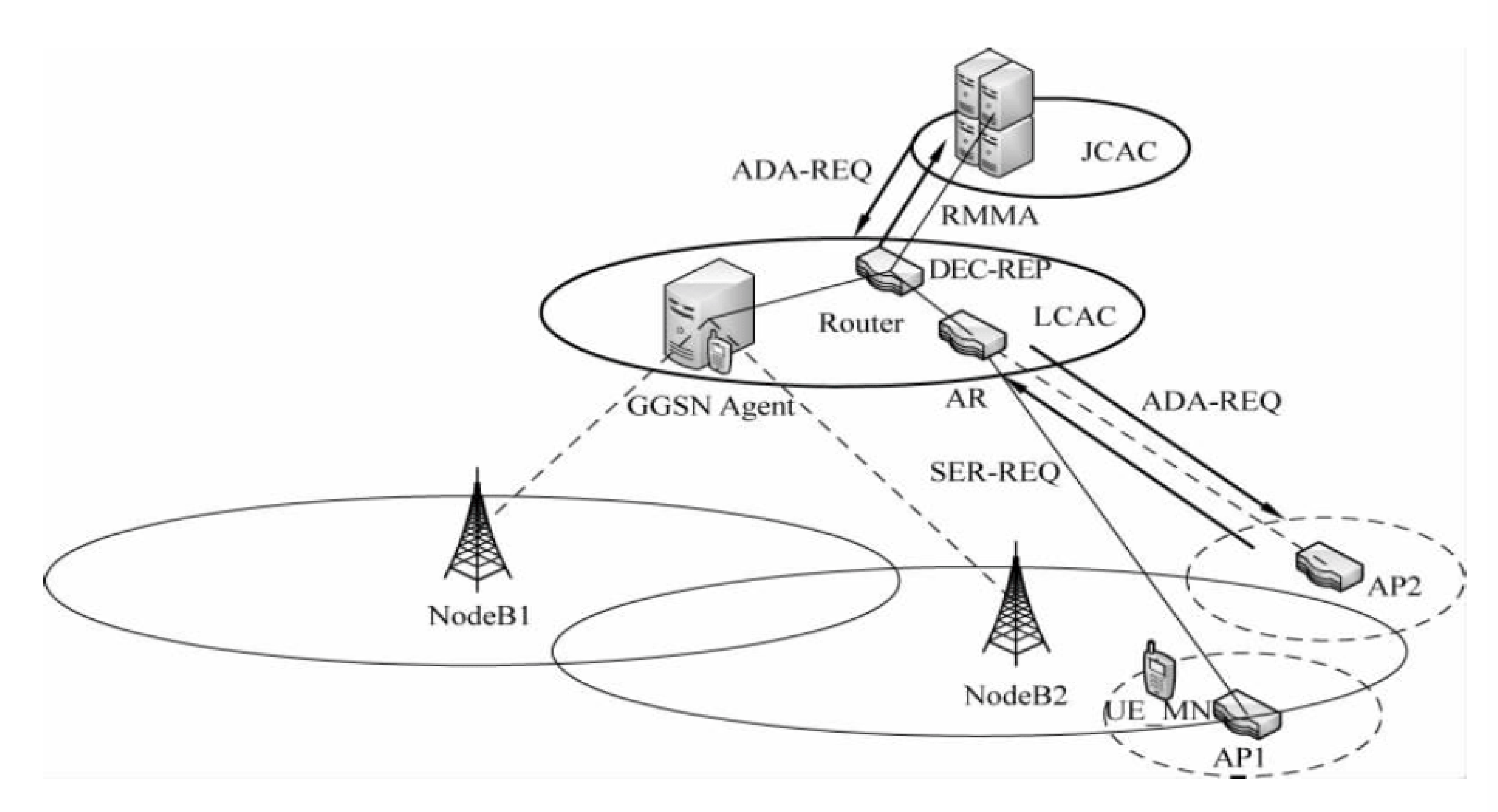


图 8-32 联合接入控制功能模块的仿真功能实体分布图

(2) LCAC 模块：本地接入控制功能模块(local call admission control)主要添加在UMTS 的 NodeB 实体或 WLAN 的 AP 实体，主要实现了网络状况实时监控模块、本地接纳控制判决模块。网络状况实时监控模块周期性收集其管辖范围内的网络状况，包括剩余带宽、已接纳连接以及当前网络负载等参数；LCAC 分别采用各自的接纳控制判决策略，进行接纳判决；将判决结果和网络性能参数封装成 DEC-REP 消息传送给 JCAC 模块。

(3) JCAC 模块：JCAC 功能实体位于 RMMA 代理上，通过修改仿真内核实现。主要实现了目标网络选择功能，考虑业务 QoS 需求以及 LCAC 功能实体上报的参数和判决信息，根据判决函数判决选择最优的网络，并向相应的 LCAC 发送业务接纳请求。

为实现不同模块之间的消息交互，仿真模块首先构建了几个新增消息类型，包括请求消息 SER_REQ、判决回复消息 DEC_RES 和自适应调整请求消息 ADA_REQ。添加消息包头(common/packet.h)PT_RRM_SOL_RMMA、PT_RRM_ACK_RMMA、PT_UMTS_REPORT。根据前面定义的请求报文消息 PT_RRM_SOL_RMMA 的格式，新包头 HDR_RRM_SOL 主要包含备选子网集结构体 node_info，它可以携带子网的 node_ip、node_type 及 power_信息，以及路由经过的 NODEB、GGSN IP 地址。

整个仿真场景为 3000m×3000m，网络拓扑分别包括 UMTS 下的接入点 NodeB 2 个、WLAN 网络下的接入点 AP 2 个，LCAC 分别在实体 GGSN 和 GMMA 上实现，router_1(1.0.0)实现 JCAC 判决功能。采用排队论模型建立了业务发生模型。假定有 10 个 MN，每个 MN 随机生成业务，业务的到达服从参数为 λ 泊松分布，业务的持续时间服从 $\mu=4$ 的指数分布。根据以上场景，主要通过三个方面验证所构建模块的可行性和准确性。首先评估 $\lambda=3$ 时不同 MN 上的业务阻塞情况，如图 8-33 所示。从中可以看出，所提出算法能够明显缓解 MN 上的业务阻塞情况，能够接纳更多的业务。

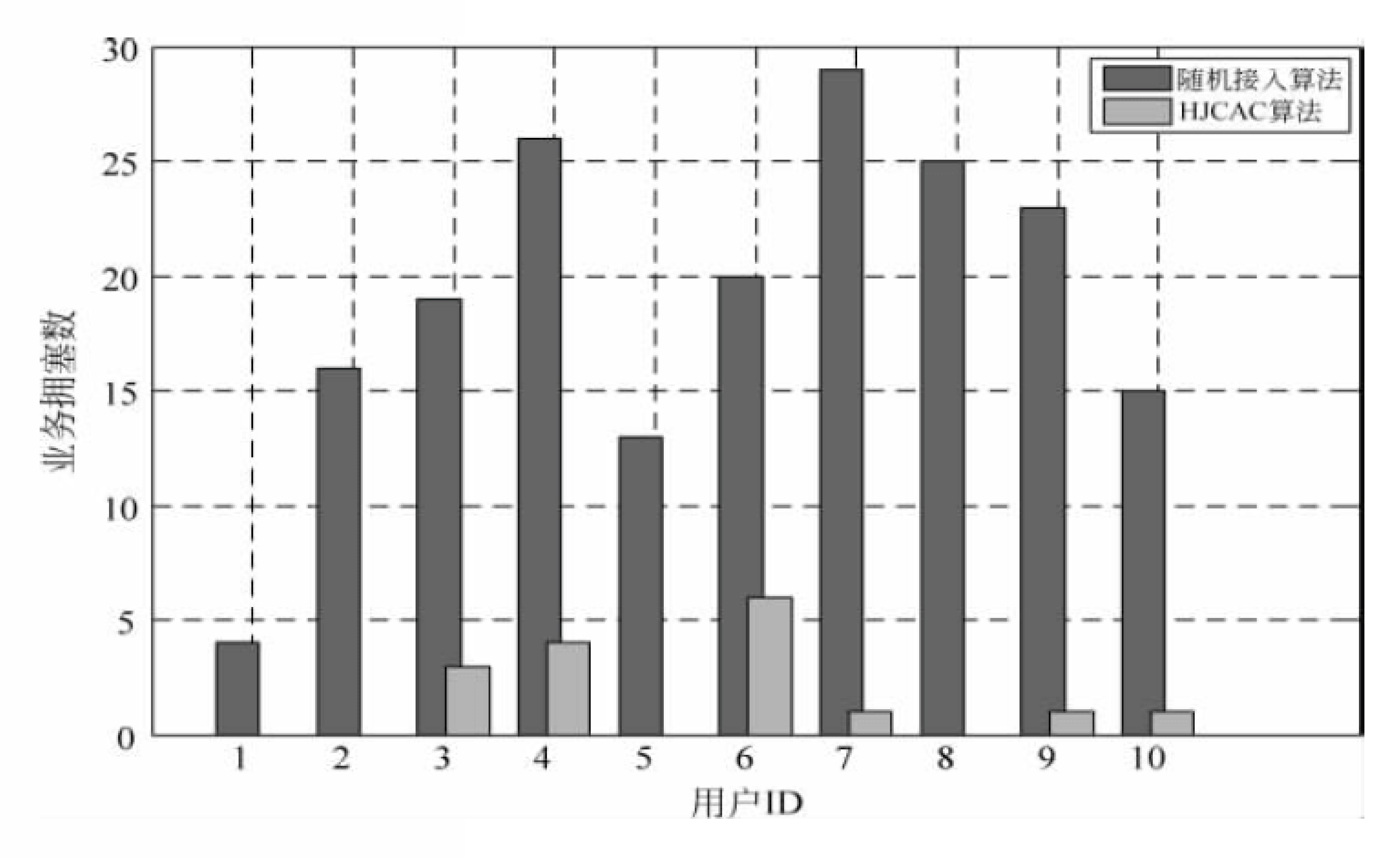


图 8-33　单个用户的业务拥塞情况

为了对不同业务生成强度下的算法正确性和可靠性进行分析，本项目接下来评估业务接纳概率随着业务生成强度 λ 的变化趋势。其结果如图 8-34 所示。从中可以看到，随着服务到达强度的不断提高，业务的接纳概率会不断降低，但是本项目所提出的分级联合接纳控制算法 HJCAC 的接纳概率均高于传统的随机接入算法。

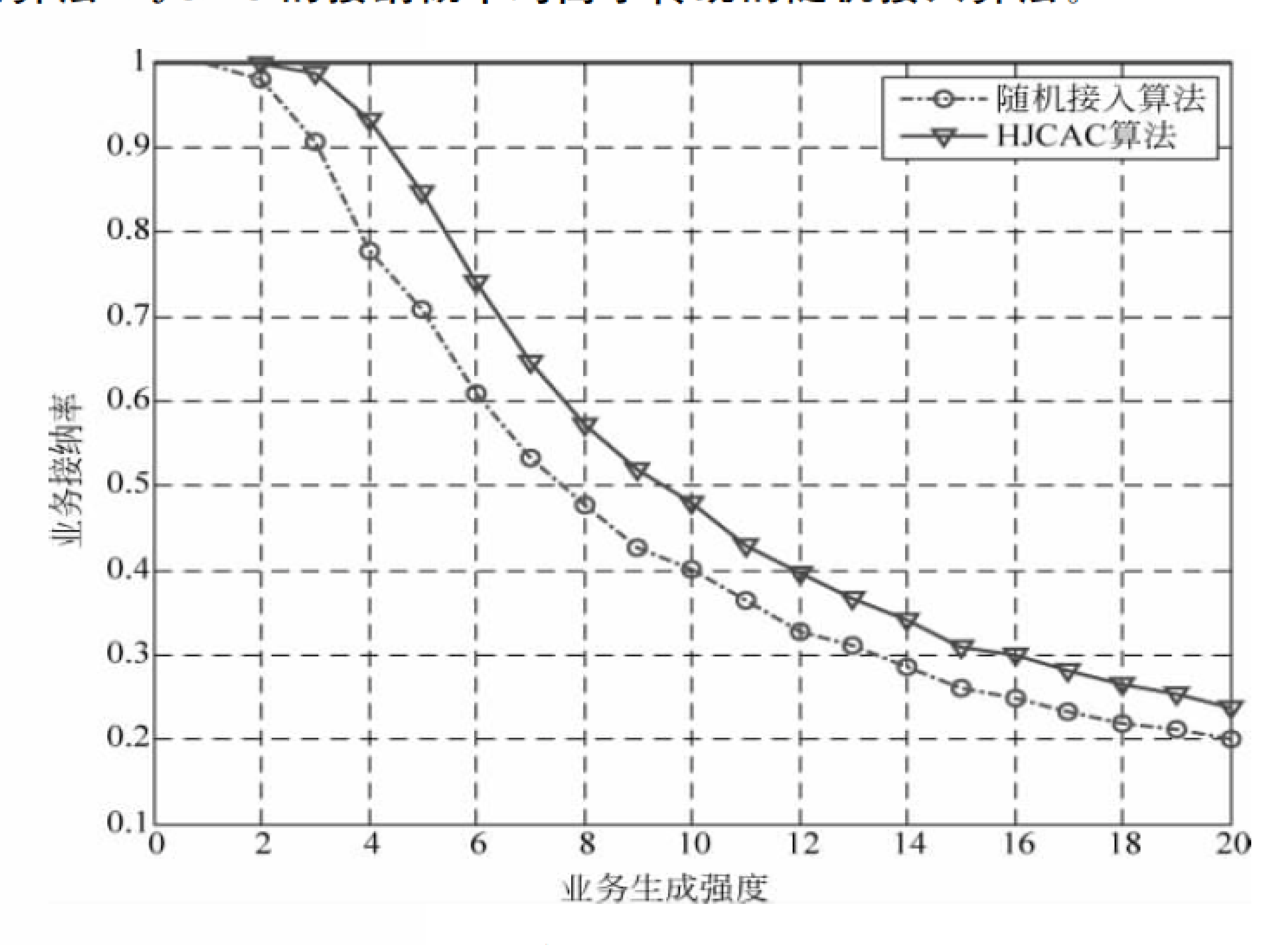


图 8-34　不同业务生成强度下接纳概率对比图

最后评估所提出方案能够为业务选择最优网络接入。这里选取了五个典型的位置点。可以看出，图 8-35 中 A 点和 C 点距离 UMTS 较近，选择 UMTS 网络；B 点因为 UMTS 负载远远重于 WLAN，最终选择了 WLAN，从而通过负载均衡，提高了网络资源的利用率；D 点和 E 点在位置上平行，但是由于网络负载情况不同，最终 D 点选择了

WLAN,而 E 点选择了 UMTS 作为接入网络。

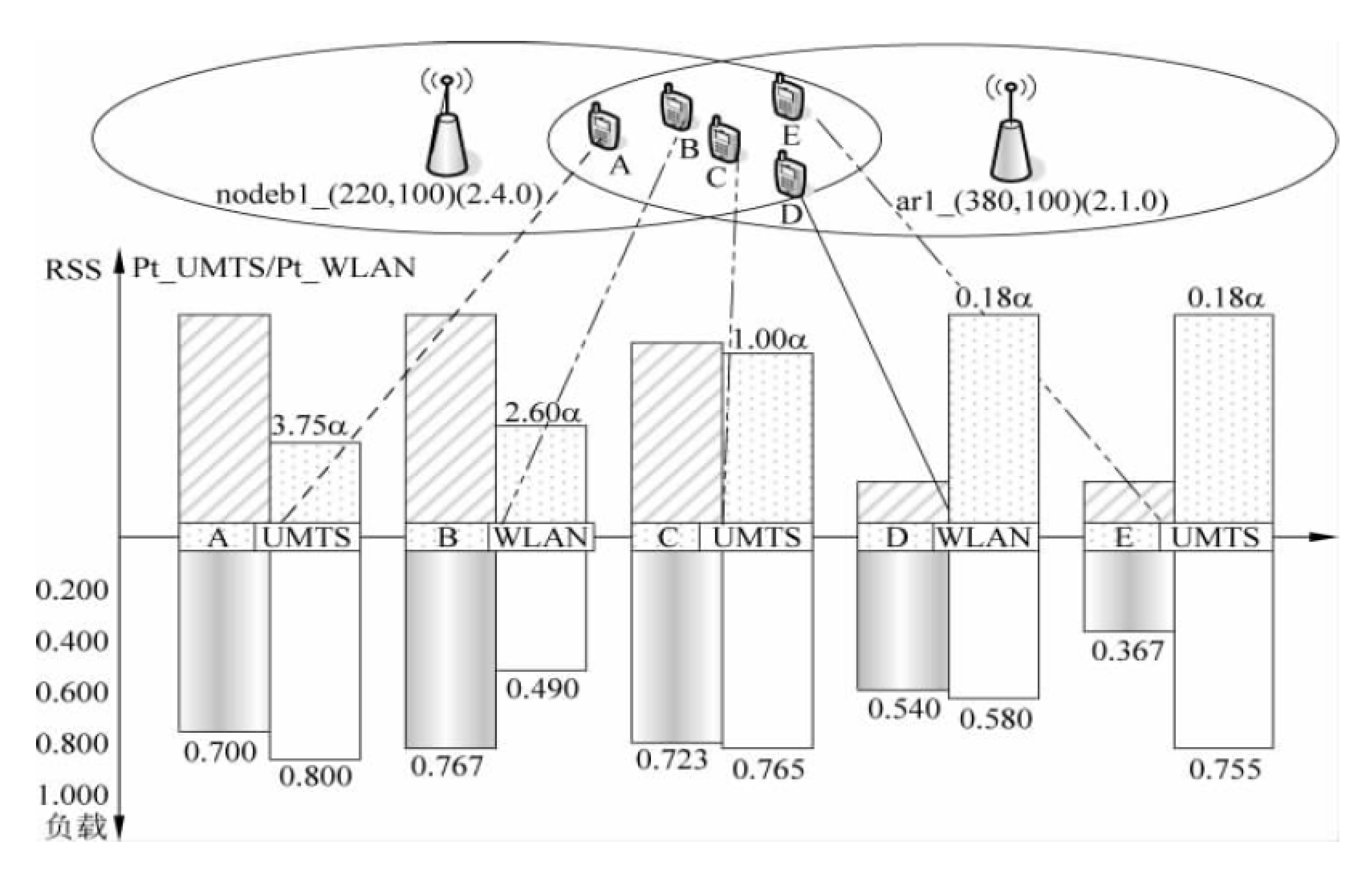


图 8-35　业务最优网络选择结果图

第9章 SDN仿真平台

软件定义网络(SDN)是一种新型的网络架构,其控制逻辑集中、转发与分离、网络可编程化等特点,给网络资源管理带来新的思路。随着SDN的快速发展,SDN的研究和仿真已成为从事SDN的研究人员所必须从事的一项工作。真实的SDN网络需要很多OpenFlow设备包括控制器、交换机、功能虚拟化等一系列设备共同依赖来搭建并应用,但因条件有限,每一位实验者不可能有这样真实的环境来验证或者实验。网络仿真软件是支持网络研究工作的工具和平台,在没有OpenFlow硬件设备下可以模拟仿真环境。本章将会介绍Mininet、OpenDaylight、FlowVisor等三款用于SDN网络研究的仿真软件,并进行仿真环境搭建及实验。

9.1 Mininet模拟网络环境

Mininet是一个虚拟网络模拟器,可以方便地在一台个人计算机或虚拟机上模拟出一个带有虚拟主机、虚拟交换机、虚拟控制器和链路。通过Mininet提供的CLI或API,用户可以方便地与网络进行交互、对网络做出调整、与他人共享甚至在真实硬件中部署。Mininet主机可以运行各种典型的Linux网络软件。由于其模拟的交换机支持OpenFlow协议,Mininet非常适合用于OpenFlow和SDN的研究开发。

9.1.1 Mininet简介

Mininet可以使用一句简单的指令在一台机器上(VM,云或者本地)迅速地创建一个真正的虚拟网络,在它之上运行真正的内核、交换机和应用程序代码。

Mininet为开发OpenFlow应用提供了一个简单而廉价的网络测设环境,支持多个开发者在同一拓扑上进行不同任务的开发,支持复杂网络结构的测试,并提供了一个能感知网络拓扑和解析OpenFlow协议的CLI交互方法,供用户调试或运行基于全网的测试。除了命令

行 CLI 的交互方式，用户还可以通过 Mininet 提供的 Python API，创建各种实验中所需的拓扑和节点结构。

由于以上的特性，在 Mininet 上开发和测试的 OpenFlow 控制器、OpenFlow 交换设备或主机，可以非常完美地移植到真实系统中以进行实地测试、性能评估和实际部署。这意味着在 Mininet 中运行良好的设计通常可以直接移植到物理交换机上进行数据包的线路转发。

可以很容易地使用 Mininet 的 CLI 和 API 与网络进行交互，自定义它、与他人分享，或将其部署在真实的硬件。因此，Mininet 对于网络的发展、教学和研究非常有帮助。

9.1.2 Mininet 安装

最简单的安装方法是下载一个已经预安装了 Mininet 的 Ubuntu 虚拟机。这个虚拟机包括 Mininet 本身并预装了所有 OpenFlow 的二进制文件和工具，并且对内核的配置进行了调整，用以支持更大的 Mininet 网络。

1. Mininet 的 VM 安装(容易，推荐)

VM 安装是安装 Mininet 最简单、最万无一失的办法，因此推荐这个安装方法。

通过如下的步骤来进行 VM 的安装：

(1) 下载 Mininet 的 VM 镜像，路径为 http://mininet.org/download/。

(2) 下载并安装一个虚拟化系统。推荐使用 VirtualBox，因为它可以自由使用在 OS X、Windows 和 Linux 操作系统中(虽然在测试中它比 VMware 稍慢)。也可以在任何操作系统中使用 QEMU，或在 Windows 或 Linux 系统中使用 VMware Workstation、在 Mac 系统中使用 VMware Fusion、在 Linux 系统中使用 KVM。

(3) 在虚拟化系统中打开 VM 镜像。

2. 本地源代码安装

这种方法适合本地虚拟机、远程 EC2 和本地安装。

此方法适合在一个未安装过 Mininet 的 Ubuntu 上进行安装。

在此推荐使用最新版本的 Ubuntu，因为它们支持新版本的 Open vSwitch。

首先需要获得源代码：

```
git clone git://github.com/mininet/mininet
```

注意，上述 git 指令将获得最新版本的 Mininet。或者可以通过如下指令检索其他的版本：

```
cd mininet
git tag  # list available versions
git checkout -b 2.2.1 2.2.1  # or whatever version you wish to install
cd ..
```

在获得了源代码以后，执行以下指令来进行安装：

```
mininet/util/install.sh [options]
```

通常的 install.sh 选项包括：

(1) -a——安装全部功能，包括 Mininet、Open vSwitch、OpenFlow Wireshark 解析器以及 POX。默认情况下，这些工具将安装在您的 home 目录创建的目录中。

(2) -nfv——安装 Mininet、OpenFlow 相关的交换机，以及 Open vSwitch。

(3) -S——在其他选项前使用该指令来放置源文件于指定的目录，而不是在 home 目录。

因此，可能会使用下列指令之一(仅一个)：

(1) 安装全部功能(使用 home 目录)。

```
install.sh -a
```

(2) 安装全部功能(使用其他的目录)。

```
install.sh -s mydir -a
```

(3) 安装 Mininet+用户的交换机+OVS (使用 home 目录)。

```
install.sh -nfv
```

(4) 安装 Mininet+用户的交换机+OVS (使用其他的目录)。

```
install.sh -s mydir -nfv
```

通过执行以下指令找到其他有用的选项(例如安装 OpenFlow Wireshark 解析器)：

```
install.sh -h
```

在安装完成之后，测试基本的 Mininet 的功能。

3. 从 Mininet 的文件包安装

如果正在使用一个新版本的 Ubuntu，可以使用 Mininet 的文件包进行安装。

首先，如果是从一个早期版本的 Mininet(像 Mininet1.0)或 Open vSwitch 升级而来，这些版本可能被编译并存储在/usr/local，因此要确保从/usr/local 删除以前任何版本的 Mininet 或 Open vSwitch 的痕迹。通过执行以下指令操作来进行：

```
sudo rm -rf /usr/local/bin/mn /usr/local/bin/mnexec \/usr/local/lib/python*/*/*
mininet* \/usr/local/bin/ovs-* /usr/local/sbin/ovs-*
```

为了确认在使用哪个 OS 版本，执行以下指令：

```
lsb_release -a
```

根据自己 Ubuntu 的版本输入以下指令来安装相对应版本的 Mininet：

```
Mininet 2.1.0 on Ubuntu 14.10: sudo apt-get install mininet
Mininet 2.1.0 on Ubuntu 14.04: sudo apt-get install mininet
Mininet 2.0.0 on Ubuntu 12.04: sudo apt-get install mininet/precise-backports
```

在这之后，需要停用 openvswitch-controller，这样 Mininet 在启动的时候可以使用其他的控制器：

```
sudo service openvswitch - controller stop
sudo update - rc.d openvswitch - controller disable
```

之后，可以通过以下指令创建 Mininet 网络并测试：

```
sudo mn  -- test pingall
```

4. 对已有的 Mininet 进行升级

如果没有对 Mininet 进行过任何改动，则可以执行以下指令来对其进行升级：

```
cd mininet
git fetch
git checkout master  #  Or a specific version like 2.2.1
git pull
sudo make install
```

也可以执行 sudo make develop 来替代 sudo make install，这将从/usr/Python/…向源树创建链接。

在进行完安装以后，通过使用很简单的指令就可以创造一个网络。例如：

```
sudo mn
```

这样便创建了一个拥有最小化拓扑的网络，包括一台控制器(controller)、一台交换机(switch)、两台主机(host)，并且进入到 CLI 界面，可以对 Mininet 创建的网络进行操作，这些内容将在下一节详细介绍。

9.1.3 Mininet 使用说明

在本节将演示 Mininet 的各种指令，并进行解释说明。

1. 查看信息

1）查看全部节点

```
mininet > nodes
```

2）查链路信息

```
mininet > net
```

3）输出各节点的信息

```
mininet > dump
```

4）对节点进行单独操作

如果想要对某个节点的虚拟机单独进行指令操作，也十分简单，格式为 node cmd。例如，查看交换机 s1 上的网络信息，只需执行：

```
mininet > s1 ifconfig
```

2. 主机之间的链接测试

由主机 h1 ping 主机 h2 的功能可以由如下指令完成：

```
mininet > h1 ping - c 6 h2
```

其中，ping -c 6 表示 ping 6 个包。

运行结果如下：

```
mininet> h1 ping h2 -c 6
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_req=1 ttl=64 time=0.037 ms
64 bytes from 10.0.0.2: icmp_req=2 ttl=64 time=0.036 ms
64 bytes from 10.0.0.2: icmp_req=3 ttl=64 time=0.048 ms
64 bytes from 10.0.0.2: icmp_req=4 ttl=64 time=0.037 ms
64 bytes from 10.0.0.2: icmp_req=5 ttl=64 time=0.084 ms
64 bytes from 10.0.0.2: icmp_req=6 ttl=64 time=0.037 ms

--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4996ms
rtt min/avg/max/mdev = 0.036/0.046/0.084/0.018 ms
```

对全网络主机互 ping 的指令是 pingall，执行该指令会对所有主机互相之间进行 ping 连通测试。

运行结果如下：

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (0/12 lost)
mininet>
```

需要了解 OpenFlow 所控制的流量。第一个主机为了向第二个主机的 MAC 地址发送 ARP 会发送一个 packet_in 消息给控制器。然后控制器发送 packet_out 消息来泛洪广播报文到交换机的其他端口。第二个主机收到 ARP 的请求并发送 RARP 回复，该回复会通过控制器发送给第一个主机，然后将其推送到每一个流表项。

3. 清除

如果 Mininet 由于某些原因崩溃，则通过以下指令将其清除：

```
sudo mn - c
```

4. 改变拓扑的大小和类型

Mininet 中通过--topo 指令可以改变网络的拓扑结构。几种拓扑类型如下：

(1) tree，n，m——第一个参数为深度，第二个参数为扇出系数。例如，执行以下指令：--tree，depth=2，fanout=8，将创建 9 个交换机连接 64 个主机的拓扑结构。

(2) single，n——单个交换机，n 个主机。

(3) linear，n——线性拓扑，n 个交换机，每个交换机下面连接一个主机。

还可以通过--link 指令改变网络中链路的状况：

```
Link: --link=tc,bw=10M,delay=10ms,loss=5%,max_que_size=5
```

以上指令表示链路的带宽为 10Mb/s，延迟为 10ms，数据包丢失率为 5%，最大排队长度为 5。

5. 自定义拓扑结构

拓扑结构也可以通过 Python 接口来进行自定义，在 custom/topo-2sw-2host.py 中给出了一个简单的例子。

这个例子中包含两个交换机，每个交换机与一个主机相连。可以通过如下指令来建立自定义拓扑结构的 Mininet 网络：

```
sudo mn --custom ~/mininet/custom/topo-2sw-2host.py --topo mytopo
```

6. 自定义 mac 地址

在默认的 Mininet 网络中，主机的 mac 地址随机分配，这使得调试变得困难。因为每一次创建 Mininet，节点的 mac 地址都会变化，这使得控制器和特定主机之间的通信变得困难。因此可以通过：

```
sudo mn --mac
```

指令配置 mac 地址。该指令使得主机和交换机的 mac 地址与其 ID 相匹配，因此很容易通过 mac 地址查找到相对应的主机和交换机。

7. 指定交换机

Mininet 中通过--switch 指令可以指定网络的交换机，例如使用 Open vSwitch 的指令如下：

```
sudo mn --switch ovsk
```

8. 指定控制器

Mininet 中通过--controller 指令可以指定网络所使用的控制器，如 Floodlight、OpenDaylight 等。指令如下：

```
--controller=remote,ip=[控制器 IP],port=[控制器监听端口]
```

其他常用 CLI 指令如表 9-1 所示。

表 9-1　Mininet 常用 CLI 指令

CLI 指令	功　能
gterm	指定节点上开启 gnome-terminal
xterm	指定节点上开启 xterm
intfs	列出所有的网络接口
iperf	两个节点之间进行简单的 iperf TCP 测试

续表

CLI 指令	功　　能
iperfudp	两个节点之间用指定带宽 udp 进行测试
noecho	运行交互式窗口，关闭回应(echoing)
pingpair	在前两个主机之间互 ping 测试
source	从外部文件中读入命令
dpctl	在所有交换机上用 dpctl 执行相关命令，本地为 tcp 127.0.0.1:6634
link	禁用或启用两个节点之间的链路
py	执行 python 表达式
sh	运行外部 shell 命令
quit/exit	退出

9.1.4　Mininet 应用实例

在现实中如果部署 SDN 网络将需要许多 OpenFlow 设备，如控制器、交换机等，同时还需要其他的一些功能来共同搭建网络并实现应用。但是由于条件受限，不可能每一个人都可以在这样真实的网络环境下进行实验。

所以在这一节将会介绍在没有 OpenFlow 硬件设备下通过 Mininet 模拟基于不同数据中心的网络拓扑。使用 Mininet 模拟不同网络数据中心拓扑可以分析网络的总流量，除此之外，还可以通过负载均衡策略来保证数据中心的可用性。本节主要基于胖树(fat-tree)拓扑结构来进行不同数据中心网络拓扑管理设计，胖树网络的好处是具有多层次的树状拓扑结构固有的容错能力。

本实验网络仿真采用了三层网络架构。三层网络架构采用层次化模型设计，即将复杂的网络设计分成几个层次，每个层次着重于某些特定的功能，这样就能够使一个复杂的大问题变成许多简单的小问题。三层网络架构设计的网络有三个层次：核心层(网络的高速交换主干)、汇聚层(提供基于策略的连接)、接入层(将工作站接入网络)。[1] 在本实验中，在各层部署交换机来实现本层的功能，并在接入层下连接主机，以模拟更加现实的网络拓扑结构。

1. 实验内容

本实验使用 Mininet 设计模拟一个多数据中心网络，主要使用 Mininet 中的 OpenFlow 交换机和控制器进行测试，模拟真实数据中心网络的参数并评估其网络性能。

一个简单的基于两个数据中心的网络拓扑如图 9-1 所示。

图 9-1 中，A1 和 A2 为核心交换机，B1～B4 为聚合交换机，C1～C4 为接入交换机，h1～h8 为主机。本实验主要测试及验证各个交换机下主机间的连通性以及主机间通信收发数据包的速度。Mininet 中的 iperf 性能测试工具可以测试不同主机间通信的带宽质量，在本实验中，主要是对相同接入交换机下的主机、相同汇聚交换机下不同接入交换机的主机、相同核心交换机不同汇聚交换机下的主机间的性能进行测试。

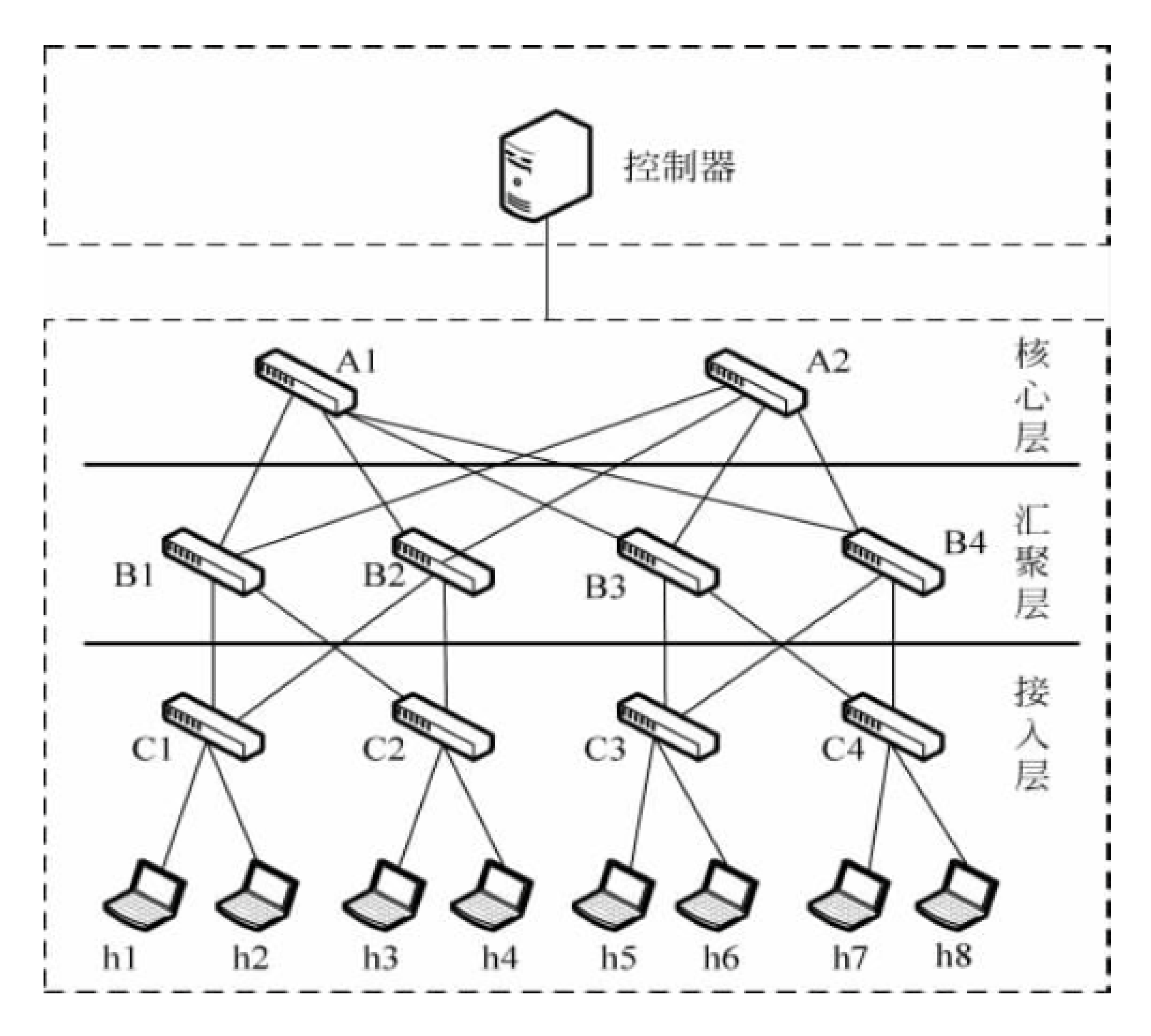


图 9-1 基于两个数据中心的网络拓扑结构

2. 实验代码

进入/home/mininet/custom 目录中，通过 Python 指令编写自定义拓扑结构，创建一个简单的基于两个数据中心的网络拓扑：

```
cd /home/mininet

vim fattree.py

#!/usr/bin/python
# encoding: utf-8
#创建网络拓扑

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import RemoteController,CPULimitedHost
from mininet.link import TCLink
from mininet.util import dumpNodeConnections

class MyTopo( Topo ):

    def __init__( self ):

        Topo.__init__( self )
        L1 = 2
        L2 = L1 * 2
```

```
        L3 = L2
        c = []
        a = []
        e = []

        # 添加核心交换机
        for i in range( L1 ):
                sw = self.addSwitch( 'c{}'.format( i + 1 ) )
                c.append( sw )

        # 添加汇聚交换机
        for i in range( L2 ):
                sw = self.addSwitch( 'a{}'.format( L1 + i + 1 ) )
                a.append( sw )

        # 添加接入交换机
        for i in range( L3 ):
                sw = self.addSwitch( 'e{}'.format( L1 + L2 + i + 1 ) )
                e.append( sw )

        # 建立核心、汇聚交换机间链接
        for i in range( L1 ):
                sw1 = c[i]
                for sw2 in a[i/2::L1/2]:
                    self.addLink( sw2,sw1 )

        # 建立汇聚、接入交换机间链接
        for i in range( 0,L2,2 ):
                for sw1 in a[i:i+2]:
                for sw2 in e[i:i+2]:
                    self.addLink( sw2,sw1 )

        # 建立主机和接入交换机间链接
        count = 1
        for sw1 in e:
                for i in range(2):
                    host = self.addHost( 'h{}'.format( count ) )
                    self.addLink( sw1,host )
                    count += 1

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

创建拓扑以后，启动 Mininet 生成具有以上拓扑结构的网络：

```
sudo mn --custom ~/mininet/custom/fattree.py --topo mytopo --controller=remote,ip=
192.168.48.130,port=6633, --switch ovsk
```

运行结果如下：

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
a3 a4 a5 a6 c1 c2 e7 e8 e9 e10
*** Adding links:
(a3, c1) (a3, c2) (a3, e7) (a3, e8) (a4, c1) (a4, c2) (a4, e7) (a4, e8) (a5, c1)
 (a5, c2) (a5, e9) (a5, e10) (a6, c1) (a6, c2) (a6, e9) (a6, e10) (e7, h1) (e7,
h2) (e8, h3) (e8, h4) (e9, h5) (e9, h6) (e10, h7) (e10, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
*** Starting 10 switches
a3 a4 a5 a6 c1 c2 e7 e8 e9 e10
*** Starting CLI:
mininet>
```

以上所示的运行结果表示创建了一个网络，拥有 2 个核心交换机、4 个汇聚交换机、4 个接入交换机，每个接入交换机连接 2 个虚拟主机，并为相应节点之间添加逻辑链路。为了提高带宽的利用率、实现负载流量的均衡，每个核心交换机下都连接 4 个汇聚交换机，每个汇聚交换机下都连接 2 个接入交换机。还可以改变代码来实现更复杂的网络拓扑，进行更全面的实验仿真。

3．实验结果

（1）同一接入交换机下的主机间连通性及通信带宽测试。使用 iperf 工具测试主机 h1 和 h2 之间的带宽：

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['2.16 Gbits/sec', '2.16 Gbits/sec']
```

实验表明，主机 h1、h2 之间可以连通，测得 h1、h2 之间 TCP 发送数据的速率为 2.16Gb/s，接收数据的速率为 2.16Gb/s。

（2）相同汇聚交换机下不同接入交换机的主机间测试。使用 iperf 工具测试主机 h1 和 h3 之间的带宽：

```
mininet> iperf h1 h3
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['1.10 Gbits/sec', '1.11 Gbits/sec']
```

实验表明，主机 h1、h3 之间可以连通，测得 h1、h3 之间 TCP 发送数据的速率为 1.10Gb/s，接收数据的速率为 1.11Gb/s。

（3）相同核心交换机不同汇聚交换机下的主机间测试。使用 iperf 工具测试主机 h1 和 h5 之间的带宽：

```
mininet> iperf h1 h5
*** Iperf: testing TCP bandwidth between h1 and h5
*** Results: ['853 Mbits/sec', '854 Mbits/sec']
```

实验表明，主机 h1、h5 之间可以连通，测得 h1、h5 之间 TCP 发送数据的速率为 853Mb/s，接收数据的速率为 854Mb/s。

9.2 控制器 OpenDaylight

OpenDaylight 项目是软件定义网络(SDN)的开源平台，采用开放的协议，提供集中的、可编程的控制和网络设备监控。像许多其他的 SDN 控制器一样，OpenDaylight 支持 OpenFlow 协议，以及提供 ready-to-install 网络的解决方案作为其平台的一部分。

在当前，大多数网络已经被设计来适应现有的需求和工作负载，SDN 可以优化现有网络来满足新的需求。面对 SDN 型网络，大家需要合适的工具帮助自己管理基础设施，这正是 OpenDaylight 的专长。OpenDaylight 是一套以社区为主导的开源框架，旨在推动创新实施以及软件定义网络(SDN)透明化。OpenDaylight 建立了一个共同的平台，可以以任何方式来配置网络并解决所面临的挑战。作为项目核心，OpenDaylight 拥有一套模块化、可插拔且极为灵活的控制器，这使其能够被部署在任何支持 Java 的平台之上。这款控制器中还包含一套模块合集，能够执行需要快速完成的网络任务。ODL 整合了开放源代码、开放标准和开放的 API 来提供一个 SDN 平台，用以加大网络的可编程化、智能化，并提高网络的适应性。

9.2.1 OpenDaylight 简介

1. OpenDaylight 结构

OpenDaylight 是一个开源的项目，它的核心是一个模块化的、可扩展的、灵活的控制平台。

从高层次看，OpenDaylight 的结构通常是层次式的。包括如下几层：

(1) 网络应用程序和业务流程。顶层包括了利用网络进行正常的网络通信的应用程序，也包括控制和监控网络行为的业务和网络逻辑应用。

(2) 控制器平台。网络架构的中间层，SDN 的抽象表示；为应用层提供了一组常用的 API(通常称为北向接口)，并且为网络中物理硬件的命令和控制部署了一个或多个协议(通常称为南向接口)。

(3) 物理和虚拟网络设备。网络架构的底层，包括物理和虚拟设备、交换机、路由器等，组成了网络中的所有节点之间的连接结构。

2. OpenDaylight 框架

OpenDaylight 的框架如图 9-2 所示。

如图 9-2 所示，OpenDaylight 支持 OSGi 框架和双向 REST 的北向 API。

OSGi 框架作为应用来使用，它和控制器运行在相同的地址空间。而 REST(基于网络的)API 虽然也作为应用来使用，但是它并不和控制器运行在同一个地址空间(甚至不一定在同一台机器上)。业务的逻辑和算法保留在应用程序中。这些应用程序使用控制器来收集网络信息，运行算法来进行分析，并使用控制器来编排新规则。控制器平台本身包含了一系列的动态可扩展的模块来执行网络所需要的任务，像一系列网络的基础服

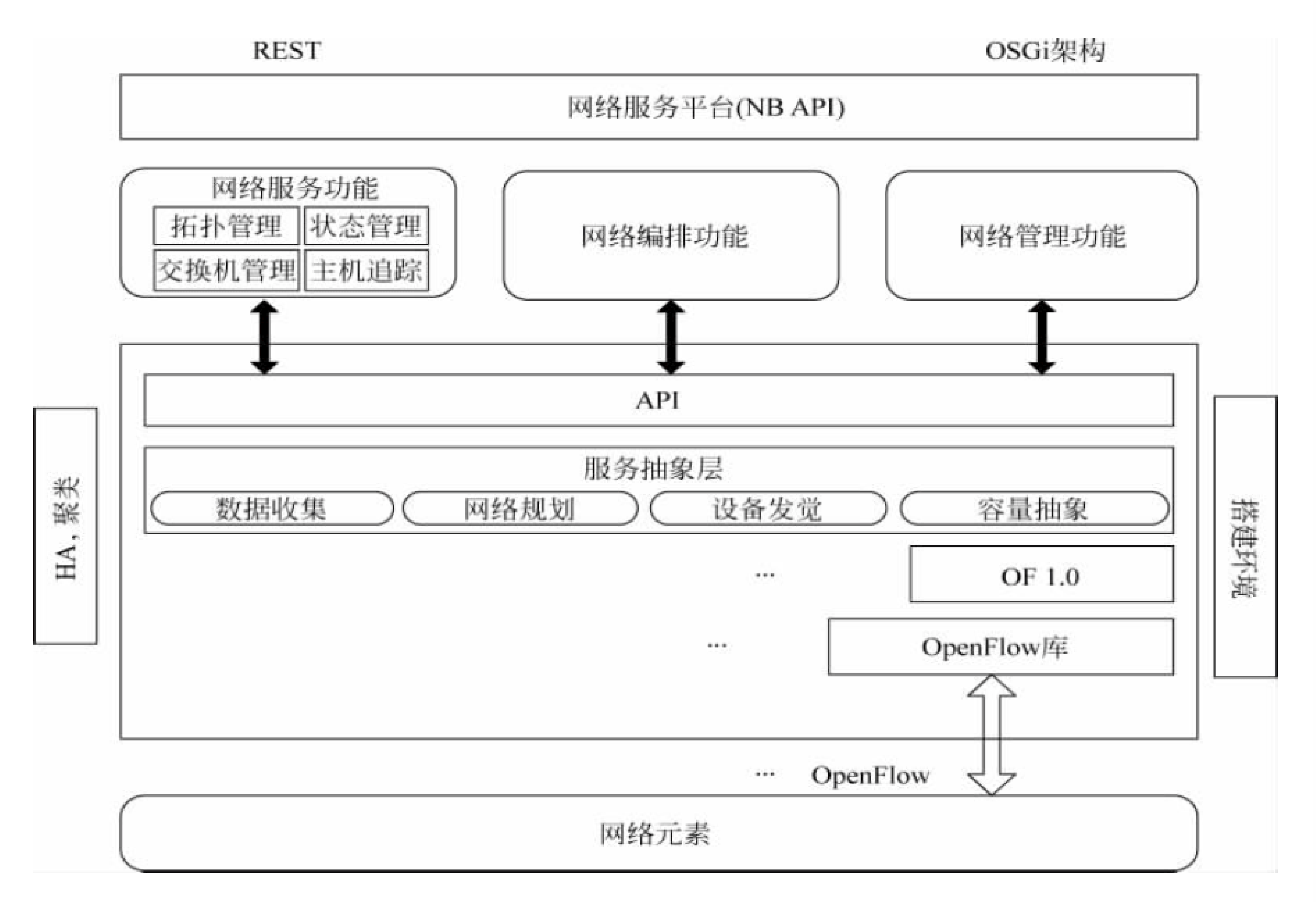


图 9-2　OpenDaylight 的框架

务,如弄清哪些设备包含在网络中,以及这些设备的容量,收集统计信息等。此外,平台化的服务和其他扩展程序也可以插入控制器平台以使得 SDN 功能增强。

南向接口能够支持多种协议(作为单独的插件),如 OpenFlow 1.0、OpenFlow 1.3、BGP-LS,这些模块动态链接到一个服务抽象层(SAL)。SAL 暴露设备服务给它北向的模块。SAL 决定在控制器和网络设备之间如何完成所请求的服务而不考虑底层协议。

控制器在其网络中控制设备,它需要知道设备的功能、可达性等,此信息由拓扑管理者进行存储和管理。其他部件如 ARP 处理程序、Host Tracker、设备管理和交换机管理为拓扑管理生成拓扑数据。

3. OpenDaylight 功能

OpenDaylight 提供了一个接口来帮助我们迅速和智能地连接网络设备以优化网络性能。

我们需要明白使用 OpenDaylight 来配置网络环境并不是一个单一的软件安装的过程,我们不仅需要安装 OpenDaylight,而且还额外地安装了像 Karaf 的功能包来满足特定的需求。

OpenDaylight 可以实现以下功能:

(1) 对网络中的物理和虚拟设备进行逻辑上的集中可编程化控制。

(2) 使用标准化、开放的协议来控制设备。

(3) 提供更高级别的抽象的能力,因此有经验的网络工程师和开发人员可以自定义网络设置和管理创建新的应用程序。

（4）集中式网络监控、管理和业务流程。

（5）主动的网络管理与流量工程。

（6）链路中的数据包通过不同的 VMs，即业务功能链（SFC）。SFC 可以实现网络功能虚拟化（NFV）。

（7）云——管理虚拟层和虚拟层下面的物理层。

下面是一些在使用 OpenDaylight 时的基本概念和工具。

1）Apache Karaf

Apache Karaf 提供了方便的安装功能，当想部署 Karaf 功能时进行安装，并且其中包含了 OpenDaylight 平台软件。默认情况下，OpenDaylight 没有预先安装的功能。

在安装 OpenDaylight 后，可以使用 Karaf 控制台安装所需要的功能来扩展网络的能力。

下面的 Karaf 功能列表中包含当创造网络环境时最有可能使用的功能。

```
install odl - alto - all
```

2）dlux 网络接口

OpenDaylight 提供 dlux 网络接口来管理网络。它的名字是 odl-dlux-core。

dlux 从 OpenDaylight 的拓扑结构和主机数据库得到并显示以下信息：

（1）网络状况。

（2）流量统计。

（3）主机的位置。

为了在安装 OpenDaylight 后使用 dlux UI，可以在 Karaf 控制台运行：

```
install odl - dlux - core
```

3）网络嵌入式体验（NeXt）

NeXt 提供了一个以网络为中心的拓扑用户界面。它可以提供以下可视化的功能：

（1）大型复杂网络拓扑。

（2）聚合的网络节点。

（3）流量/路径/通道/组的可视化。

（4）不同的布局算法。

（5）覆盖图。

（6）预设的用户友好的交互。

4）模型驱动的服务抽象层（MD-SAL）

MD-SAL 是 OpenDaylight 框架，它允许开发者以服务和协议驱动的形式开发新的 Karaf 功能，并将其连接到另一个。MD-SAL 由以下两部分组成。

（1）一个共享的数据存储——维护如下所示的树形结构：

- 配置数据存储——用以维护所需的网络状态。
- 操作数据存储——这是一个基于管理的网络要素数据的实际网络状态表示。

（2）一条消息总线——它为各种服务和协议驱动程序的通信提供了一条路径。

如果在与 OpenDaylight 通信时使用了 dlux 或 REST API，微服务架构将允许选择

可用的服务、协议和 REST API。

本节对 OpenDaylight 的结构、架构和功能进行了简单的介绍，下面将详细介绍 OpenDaylight 的安装和使用。

9.2.2 OpenDaylight 安装配置

OpenDaylight 现在已发布 4 个版本，分别为 Hydrogen（氢）、Helium（氦）、Lithium（锂）、Beryllium（铍）。

1. ODL 安装

（1）在 ODL 官网上获取 ODL 代码：

```
git clone https://git.opendaylight.org/gerrit/p/controller.git
```

（2）编译控制器：

```
cd controller/opendaylight/distribution/opendaylight
mvn clean install
```

（3）解压获取的安装包文件，并进入解压目录：

```
unzip distribution-karaf-0.2.0-Helium.zip
cd distribution-karaf-0.2.0-Helium/
cd bin
./karaf   ##执行 karaf 文件
```

出现图 9-3 所示界面，表示安装正确。



图 9-3　安装正确弹出界面

（4）运行 ODL Controller：

```
cd controller/opendaylight/distribution/opendaylight/target/distributions.oepndaylight-
OSGIpackage/opendaylight
./run.sh
```

按以上方法，每次运行 ODL Controller 都要进入 ODL 的目录，因此为了简化流程，可以编写一个启动脚本来控制 ODL Controller 启动。

编写过程如下：

```
cd controller
vim odlStart.sh
```

输入以下内容：

```
#!/bin/bash
./opendaylight/distribution/opendaylight/target/distribution.opendaylight-osgipackage/
opendaylight/run.sh
```

编写完成之后修改其权限为可执行文件：

```
sudo chmod a+x odlStart.sh
```

这样只需要运行 odlStart.sh 文件便可以启动 ODL Controller。

2. 连接 Controller 和 Mininet

1）启动 ODL

按照上面介绍的步骤启动 ODL Controller，打开浏览器，输入 http://[ODL_host_ip]:8080（[ODL_host_ip]为安装 ODL 所在的主机 IP 地址），进入 OpenDaylight 的 Web UI 界面进行访问，用户名为 admin，密码为 admin，如图 9-4 所示。

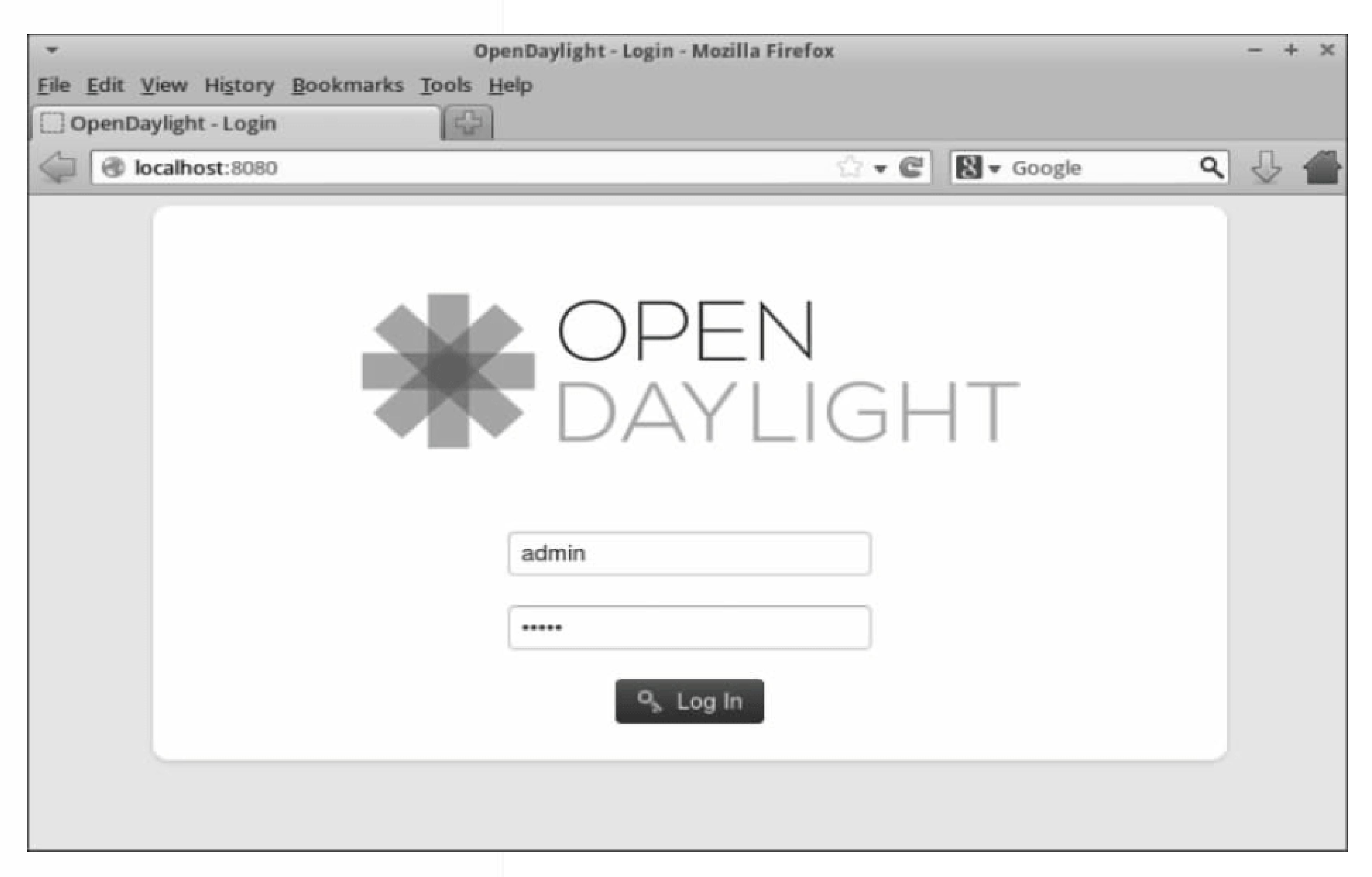


图 9-4　启动 ODL

2）启动 Mininet 连接 ODL

执行以下指令启动 Mininet 连接到 ODL Controller。在这里，采用自定义的拓扑，有两个交换机相连，并且每个交换机分别下挂两个主机。Controller 使用 remote

controller,IP 地址为 ODL 所在主机的 IP 地址,监听端口为 ODL 的监听端口。

```
sudo mn --topo mytopo --controller=remote,ip=192.168. 48.130,port=6633
```

3. Controller 功能简介

Mininet 连接上 ODL Controller 后,会在 ODL 的首页形成网络的拓扑图,此时的拓扑图只会显示交换机而不会显示主机,这是根据链路发现协议(LLDP)来决定的,当主机发起流量时,相应的主机才会显示在拓扑图中。在 Mininet 的 CLI 界面执行 pingall 指令以后,可以看到包含主机的全网络的拓扑图,如图 9-5 所示。

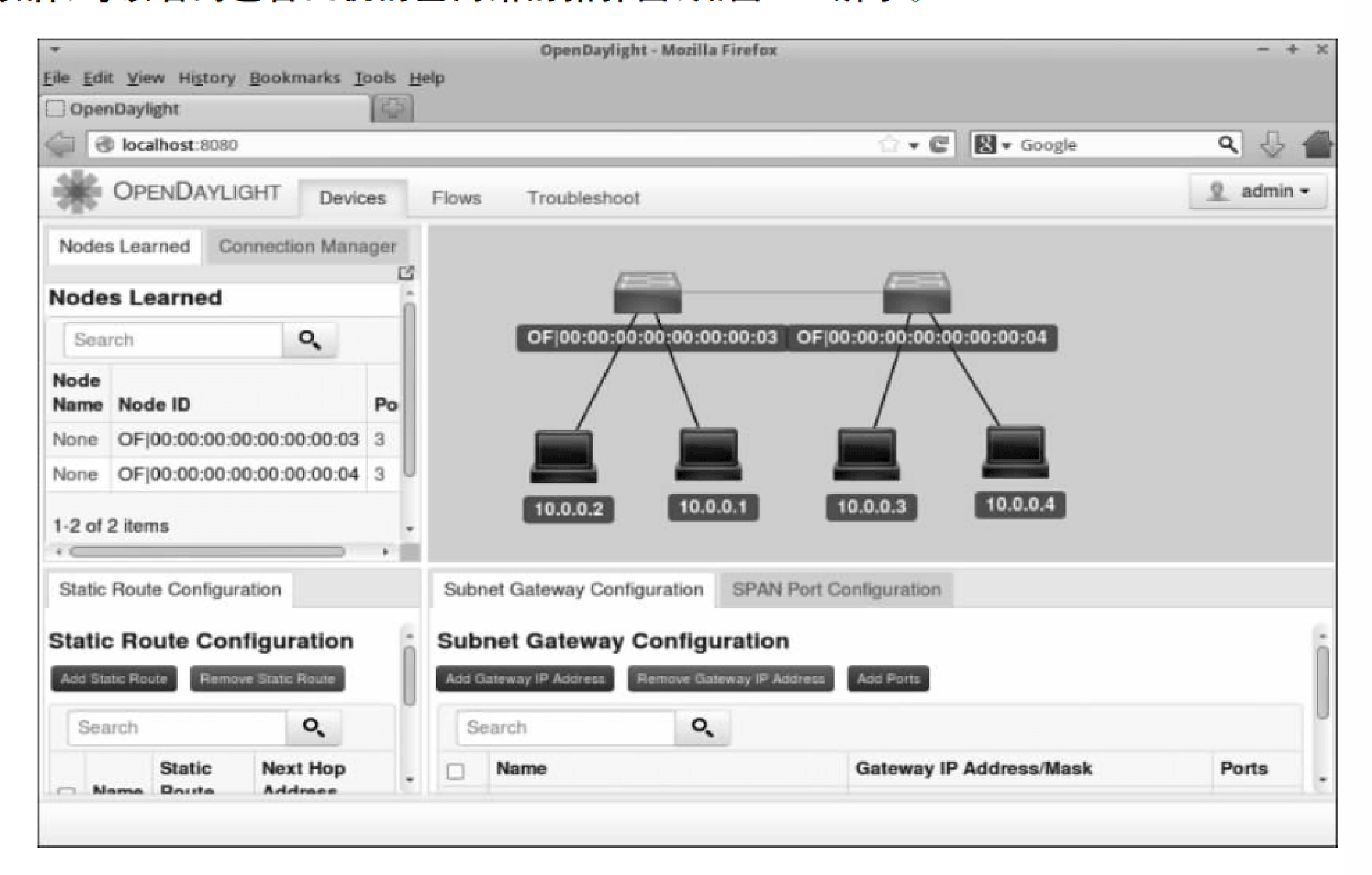


图 9-5 全网络的拓扑图

如图 9-5 所示,ODL Controller 主要包含的功能有设备、静态流表及统计。

Devices 功能可以实现网络节点的学习(Nodes Learned)、连接管理(Connection Manager)、网络拓扑图、静态路由配置(Static Route Configuration)、子网网关配置(Subnet Gateway Configuration)及 SPAN Port Configuration。

其中,Nodes Learned 功能可以发现网络设备,得到对应的网络节点(网络设备、交换机)的 ID(Node ID)及对应交换机的端口信息;Connection Manager 可以对这些网络设备进行管理,可以实现网络设备的添加和删除;Static Route Configuration 可以手动添加或删除静态路由,进行路由配置管理;Subnet Gateway Configuration 可以手动实现子网网关的添加和删除配置,并可以实现只针对交换机具体端口的网关的配置,Controller 的三层转发功能必须配置网关;SPAN Port Configuration,即交换机分析器的端口配置。SPAN 全称为 Switched Port Analyzer,直译为交换端口分析器,是一种交换机的端口镜像技术。主要是为了给某种网络分析器提供网络数据流,SPAN 并不会影响源端口的数

据交换，它只是将源端口发送或接收的数据包副本发送到监控端口。[2]

9.2.3 OpenDaylight 应用实例

基于 OpenDaylight 的二层转发原理。

1. 实验原理

在 SDN 网络中，位于底端的主机并不知道它连接到了 SDN 网络，某台主机要发送数据包到另一台主机的时候，仍然需要进行 IP 到 MAC 地址的 ARP 解析。由于在 SDN 网络中引入了控制器的角色，其二层数据转发的机制与普通二层以太交换机的洪泛加 MAC 地址学习机制存在着很大的差异。当源 IP 要与目的 IP 通信时，会将 ARP 请求上传到控制器，由控制器来询问目的主机的 MAC 地址并将结果返回给源主机。

本实验将在 SDN 网络环境中配置相同网段的两台主机，通过测试两台主机之间的数据转发过程来了解 SDN 的二层转发机制。

2. 实验任务

本实验通过 ODL Controller 与 Mininet 相结合来搭建网络测试环境，并研究 SDN 环境下二层网络转发的机制和特性。在本实验中创建的网络拓扑结构中有 1 台 ODL Controller、2 台交换机，每台交换机分别连接 2 台主机，一共 4 台主机（这 4 个主机属于同网段）。相应的网络拓扑如图 9-6 所示。

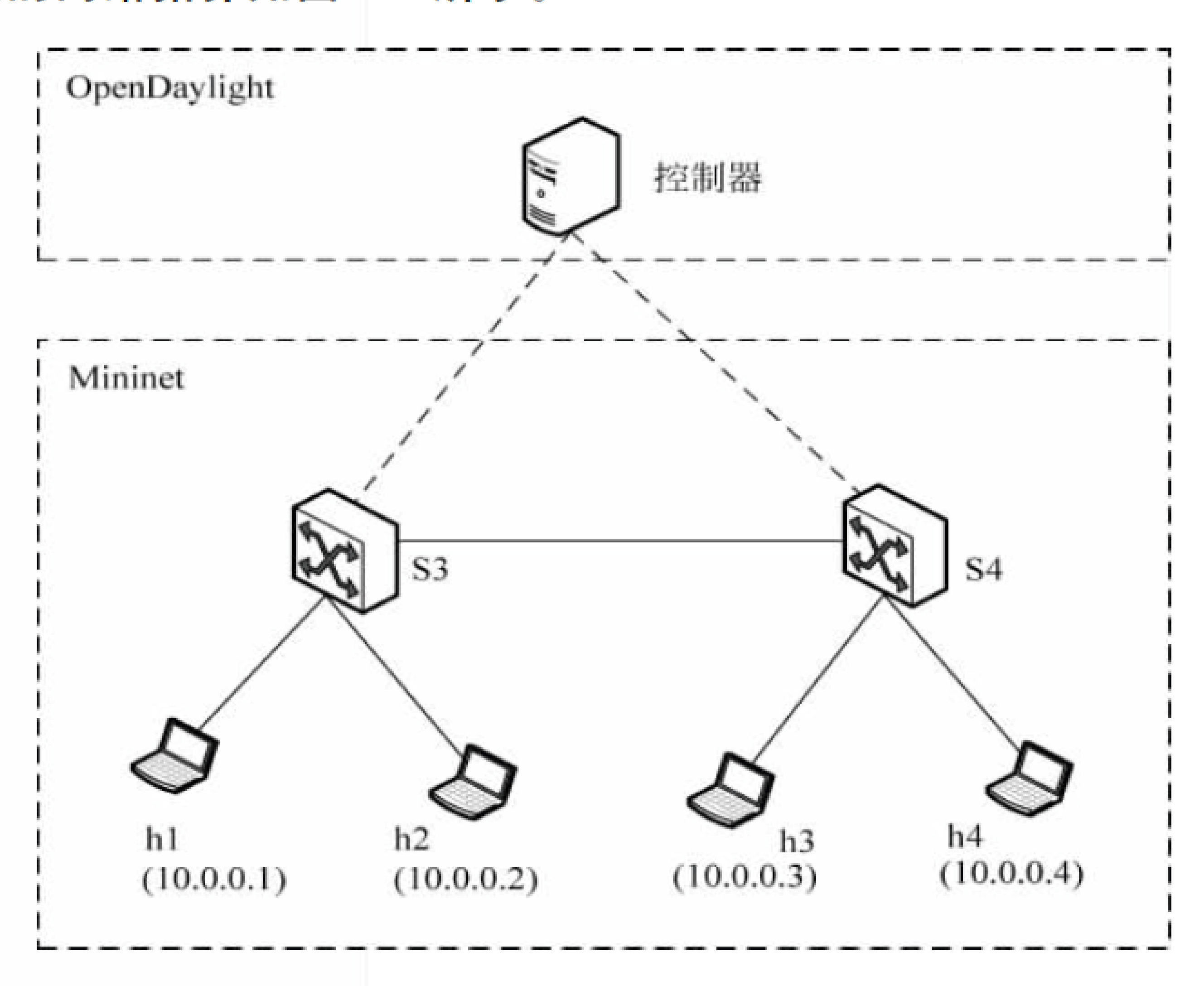


图 9-6 实验网络拓扑图

用 Python 编写测试网络的拓扑结构，并将配置文件保存在/mininet/custom 目录下的 topo-2sw_2host. py 文件内：

```
from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        leftHost = self.addHost( 'h1' )
        leftHost1 = self.addHost( 'h2' )
        leftHost2 = self.addHost( 'h3' )
        rightHost = self.addHost( 'h4' )
        leftSwitch = self.addSwitch( 's3' )
        rightSwitch = self.addSwitch( 's4' )

        # Add links
        self.addLink( leftHost1, leftSwitch )
        self.addLink( leftHost2, rightSwitch )
        self.addLink( leftHost, leftSwitch )
        self.addLink( leftSwitch, rightSwitch )
        self.addLink( rightSwitch, rightHost )


topos = { 'mytopo': ( lambda: MyTopo() ) }
```

启动 Mininet 测试环境，使用以下指令生成测试拓扑结构：

```
sudo mn --custom ~/mininet/custom/topo-2sw-4host.py --topo mytopo --controller=
remote,ip=192.168.48.130,port=6633
```

登录 ODL 的 Web 页面查看网络的拓扑如图 9-7 所示。

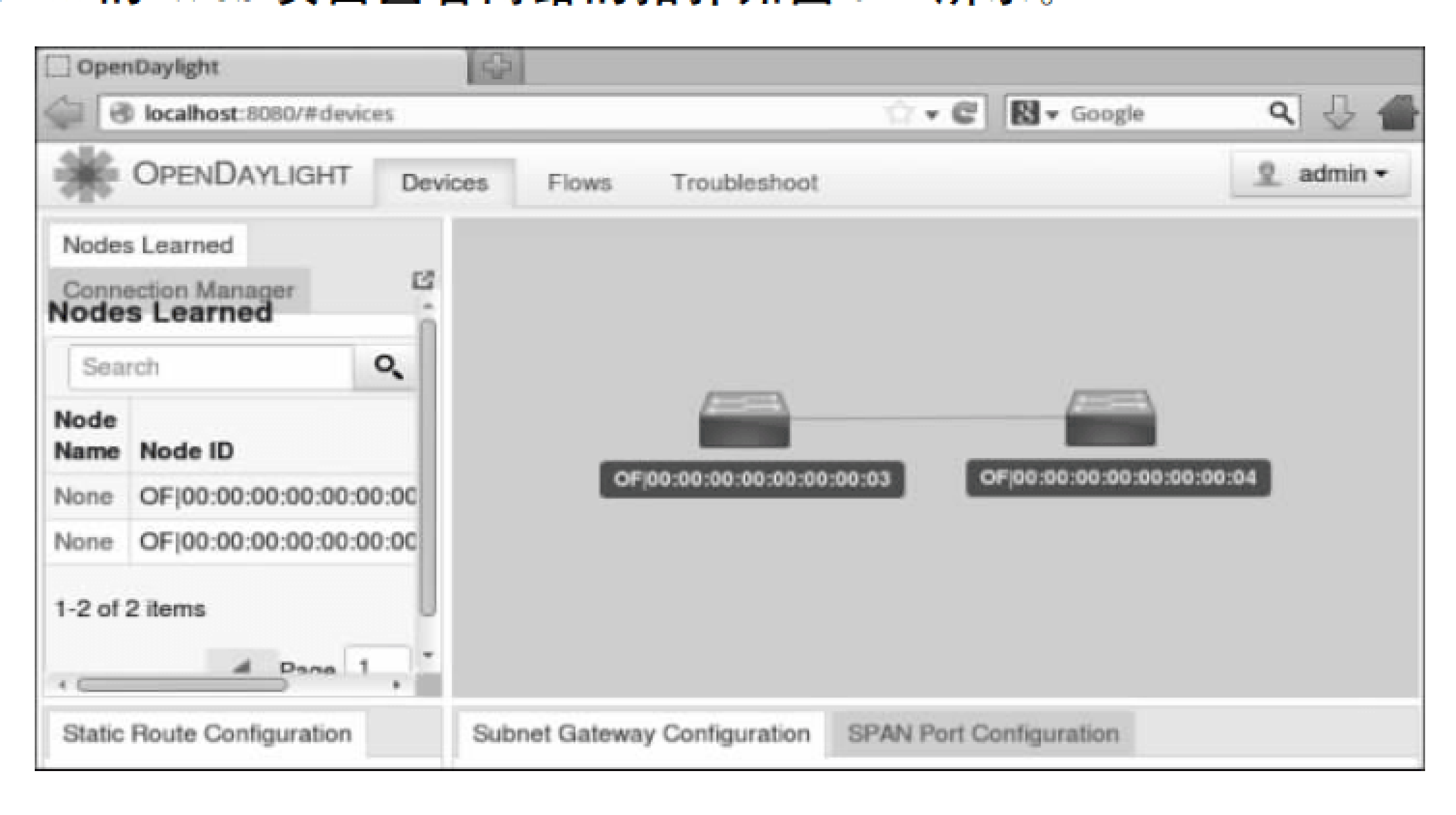


图 9-7 登录 ODL 的 Web 页面查看网络的拓扑图

下面进行抓包验证：

（1）在系统命令行执行以下指令开始抓包：

```
tcpdump -i any port 6633 -s0 -w 206.pcap
```

运行结果如下：

```
ubuntu@ubuntu:~$ sudo tcpdump -i any port 6633 -s0 -w 206.pcap
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 6553
5 bytes
```

（2）在 Mininet 命令行执行以下指令进行 h1 和 h3 的连接性测试：

```
h1 ping h3
```

（3）在 ODL 的 Web 页面查看网络的拓扑如图 9-8 所示。

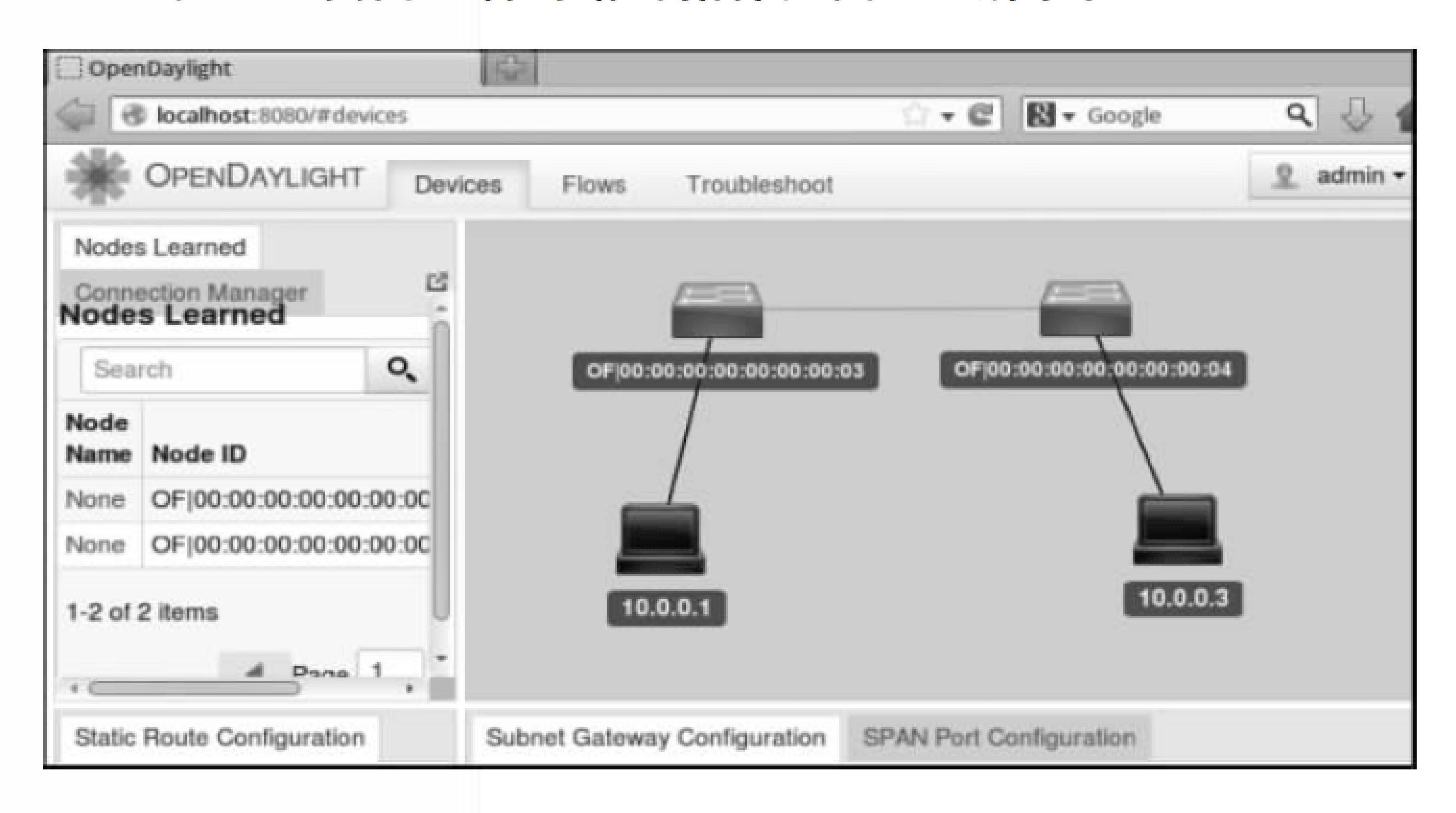


图 9-8　在 ODL 的 Web 页面查看网络的拓扑图

（4）在 ODL 的 Web 页面选择 Troubleshoot 选项卡，查看网络中的两台交换机下发的流表，如图 9-9 所示。

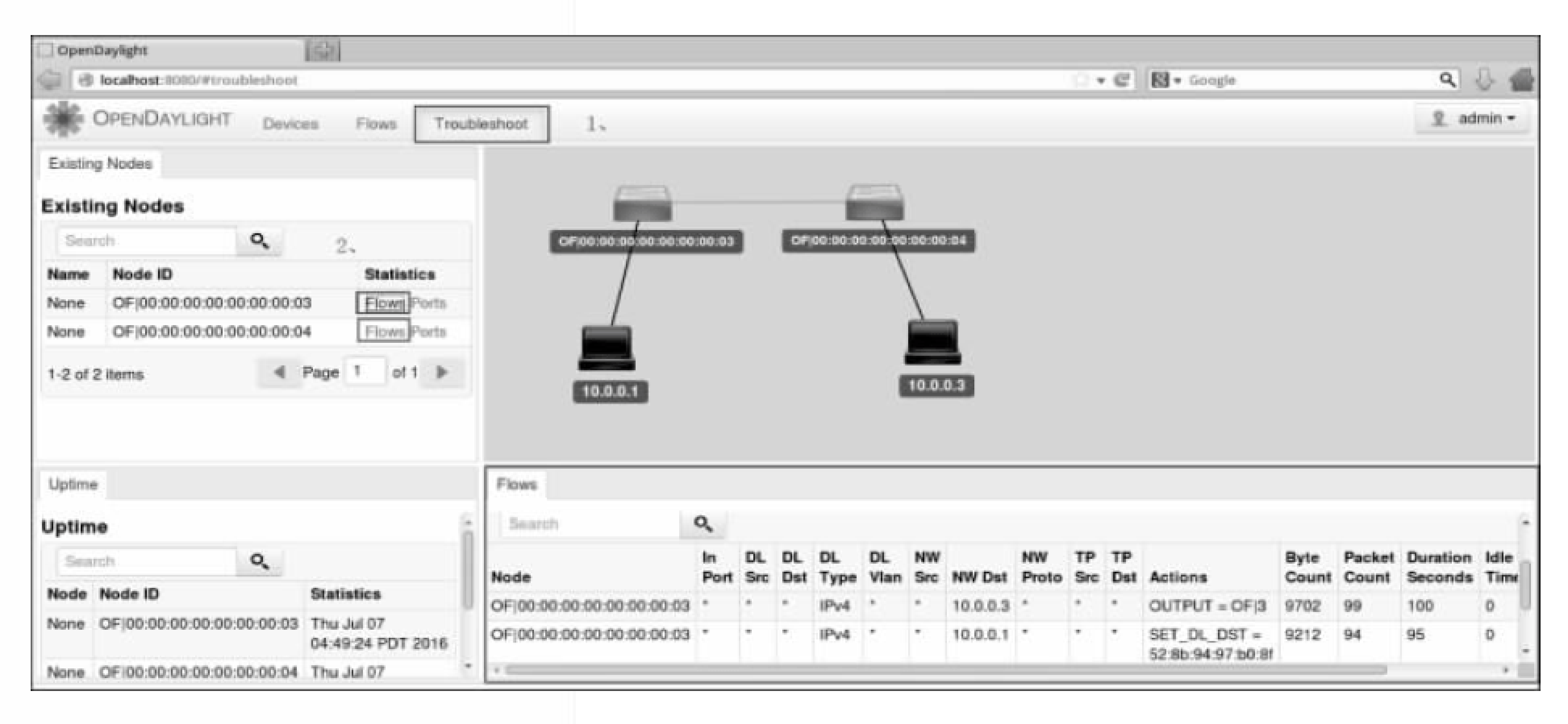


图 9-9　两台交换机下发的流表

S3 交换机上的流表如图 9-10 所示。

Node	In Port	DL Src	DL Dst	DL Type	DL Vlan	NW Src	NW Dst	NW Proto	TP Src	TP Dst	Actions	Byte Count	Packet Count	Duration Seconds	Idle Timeout	Priority
OF|00:00:00:00:00:00:00:03	*	*	*	IPv4	*	*	10.0.0.3	*	*	*	OUTPUT = OF|3	9702	99	100	0	1
OF|00:00:00:00:00:00:00:03	*	*	*	IPv4	*	*	10.0.0.1	*	*	*	SET_DL_DST = 52:8b:94:97:b0:8f OUTPUT = OF|2	9212	94	95	0	1

图 9-10　S3 交换机上的流表

S4 交换机上的流表如图 9-11 所示。

Node	In Port	DL Src	DL Dst	DL Type	DL Vlan	NW Src	NW Dst	NW Proto	TP Src	TP Dst	Actions	Byte Count	Packet Count	Duration Seconds	Idle Timeout	Priority
OF|00:00:00:00:00:00:00:04	*	*	*	IPv4	*	*	10.0.0.3	*	*	*	SET_DL_DST = de:d8:60:13:80:3c OUTPUT = OF|1	85260	870	871	0	1
OF|00:00:00:00:00:00:00:04	*	*	*	IPv4	*	*	10.0.0.1	*	*	*	OUTPUT = OF|2	84770	865	866	0	1

图 9-11　S4 交换机上的流表

（5）执行以下指令查看两台交换机上的流表内容如图 9-12 所示。

```
ovs－ofctl dump－flows s3
ovs－ofctl dump－flows s4
```

```
ubuntu@ubuntu:~$ sudo ovs-ofctl dump-flows s3
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=1149.027s, table=0, n_packets=1148, n_bytes=112584, priority=1,ip,nw_dst=10.0.0.3 actions=output:3
 cookie=0x0, duration=1144.032s, table=0, n_packets=1143, n_bytes=112014, priority=1,ip,nw_dst=10.0.0.1 actions=mod_dl_dst:52:8b:94:97:b0:8f,output:2
ubuntu@ubuntu:~$ sudo ovs-ofctl dump-flows s4
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=1149.905s, table=0, n_packets=1149, n_bytes=112682, priority=1,ip,nw_dst=10.0.0.3 actions=mod_dl_dst:de:d8:60:13:80:3c,output:1
 cookie=0x0, duration=1144.909s, table=0, n_packets=1144, n_bytes=112112, priority=1,ip,nw_dst=10.0.0.1 actions=output:2
```

图 9-12　两台交换机上的流表内容

3. 实验结果

下面结合抓包软件所得结果分析 SDN 的二层转发过程，如图 9-13 所示。

（1）源主机 h1(10.0.0.1)发出 ARP 请求解析 h3(10.0.0.3)的 MAC 地址，交换机 S3 不知道应怎样转发该请求，因此将其通过 Packet In 消息上传到 Controller 进行处理。

（2）Controller 发现此 ARP 消息由 h1(10.0.0.1)发出，它也同时得到了 h1 的位置信息。此时 Controller 可以计算网络拓扑，得到整个网络中各节点到 10.0.0.1 的转发路径。

（3）在收到 ARP 请求后，Controller 会要求每一台交换机所对应 10.0.0.0/8 网段的非交换机互联端口(只有这些端口是连接主机或传统网络的)发出 ARP 来请求 10.0.0.3 的 MAC 地址。此时 Controller 会将源 IP 改为默认网关 IP 地址，然后发出。

（4）只有 h3(10.0.0.3)才会响应此 ARP，它将 ARP Response 发送到 S4。S4 不知道如何处理，便将 ARP Response 封装在 Packet In 包中发送到 Controller。

（5）Controller 发现这是 ARP 响应，并且正是之前 10.0.0.1 发送的 ARP 请求的响应，因此它会通过 OF 协议将该 ARP 发送到 S1，并告知 S1 将此 ARP 由 h1 所在端口发出。

（6）Controller 在收到 h3 的 ARP 后也得知了 10.0.0.3 的位置，它会依据网络拓扑

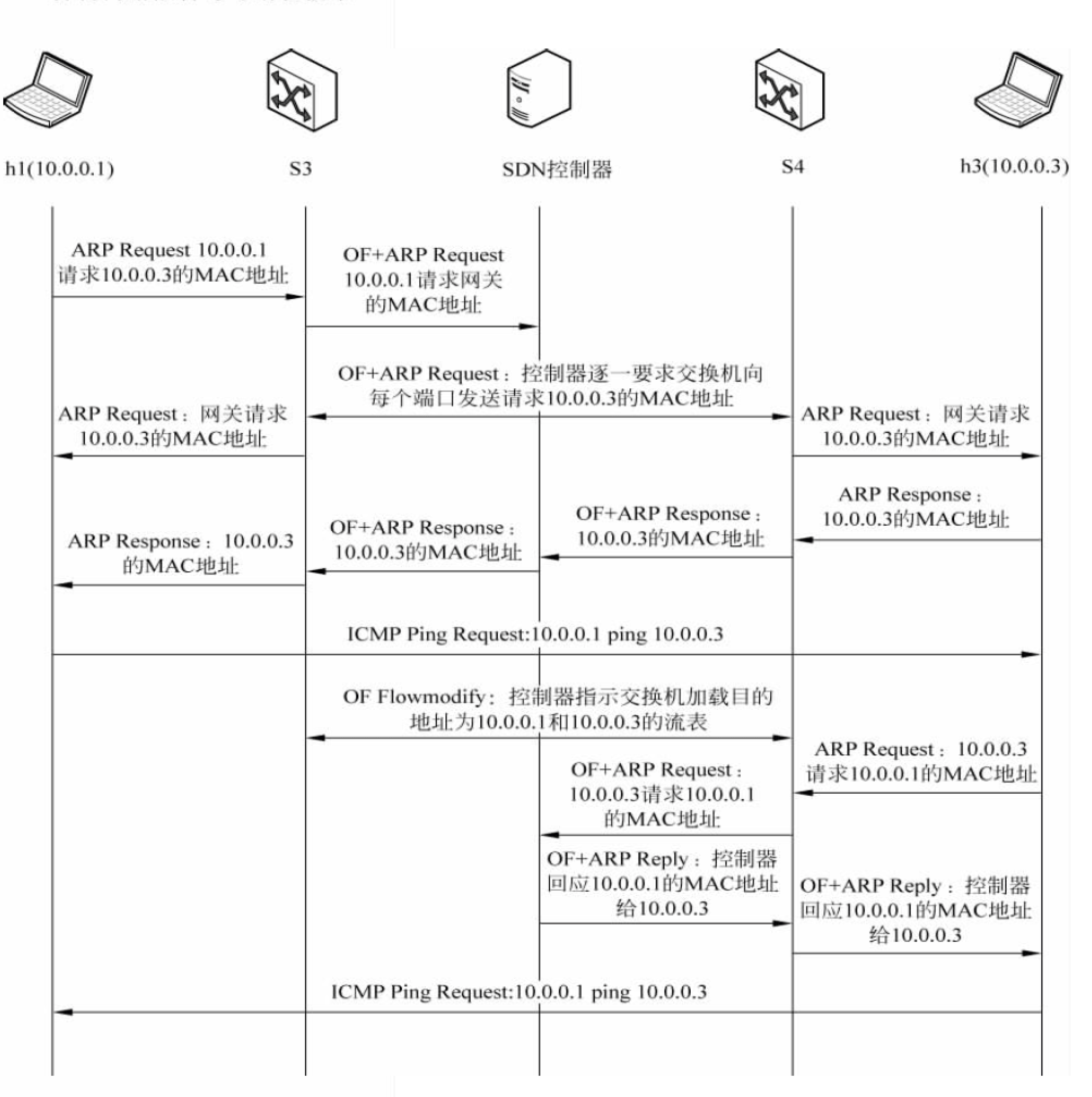


图 9-13　SDN 的二层转发过程

计算，得到全网到达 10.0.0.3 的路径，并将流表通过 OF Flow Modify 消息发送给每一台交换机。

(7) h1 收到 ARP Response 后完成 ARP 解析过程，然后它会构造 ICMP PING Request 数据包，其中源和目的 MAC 分别为 h1 和 h3 的 MAC 地址，源和目的 IP 分别为 h1 和 h3 的 IP 地址。由于 S3 和 S4 都已经获得了到 h3(10.0.0.3)的流表，因此该包会直接发送到 h3。

(8) h3 收到该 ICMP PING Request，发现其源地址是 h1，但是这时候它并没有 h1 的 MAC 地址，于是还要进行一次 ARP 解析，S4 次将 ARP 请求封装成 Packet In 消息发送到 Controller。

(9) Controller 已经知道 h1 的 MAC 地址，因此可直接响应，并通过 OF 向 SW2 返

回 ARP 结果和消息应发出的端口(h3 接入的端口)。

(10) h3 收到 ARP 后,便可以构造 ICMP Response 包,发送到 S4,S4 根据 h1 目的地址匹配转发表将其转发到 S3,S3 根据 h1 目的地址匹配转发表将其发送到 h1 对应的端口。由此 h1 到 h3 的双向转发链路完全可以通信。

9.3 网络虚拟层 FlowVisor

网络虚拟化是一种将底层网络中的硬件以及配套的软件资源进行整合,形成统一管理实体的技术,通过虚拟网络资源到物理网络资源的映射,使得多个逻辑虚拟网络共享底层物理网络基础设施。网络虚拟化技术是当今网络革新的重要技术之一。

FlowVisor 是建立在 OpenFlow 之上的网络虚拟化工具,它可以将物理网络划分成多个逻辑网络,从而实现虚网划分。它为管理员提供了通过定义流规则,而不是调整路由器/和交换机配置的方式来管理网络。

FlowVisor 在数据平面与控制平面之间添加虚拟化层,通过虚拟化层控制各个虚拟网络逻辑与物理网络设备之间的映射关系,隔离各个虚拟网络的转发表以及流量,其核心是把网络中所有的流量分类到多个“分片”,将每一个“分片”交由一个控制器进行控制。通过 FlowVisor,一个完整的 OpenFlow 网络可以划分成多个逻辑网络,每一个逻辑网络被称为一个分片(slice)。

9.3.1 FlowVisor 简介

1. FlowVisor 设计原理

就如同计算机虚拟化的效果一样,可以把一个硬件层抽象成一个逻辑层,希望让网络可以抽象出一层虚拟网络,让底层的设备傻瓜化,可以通过上层的控制端来进行对底层虚拟网络层的控制转发。

如图 9-14 所示是一个简单的 FlowVisor 和虚拟主机的对比。

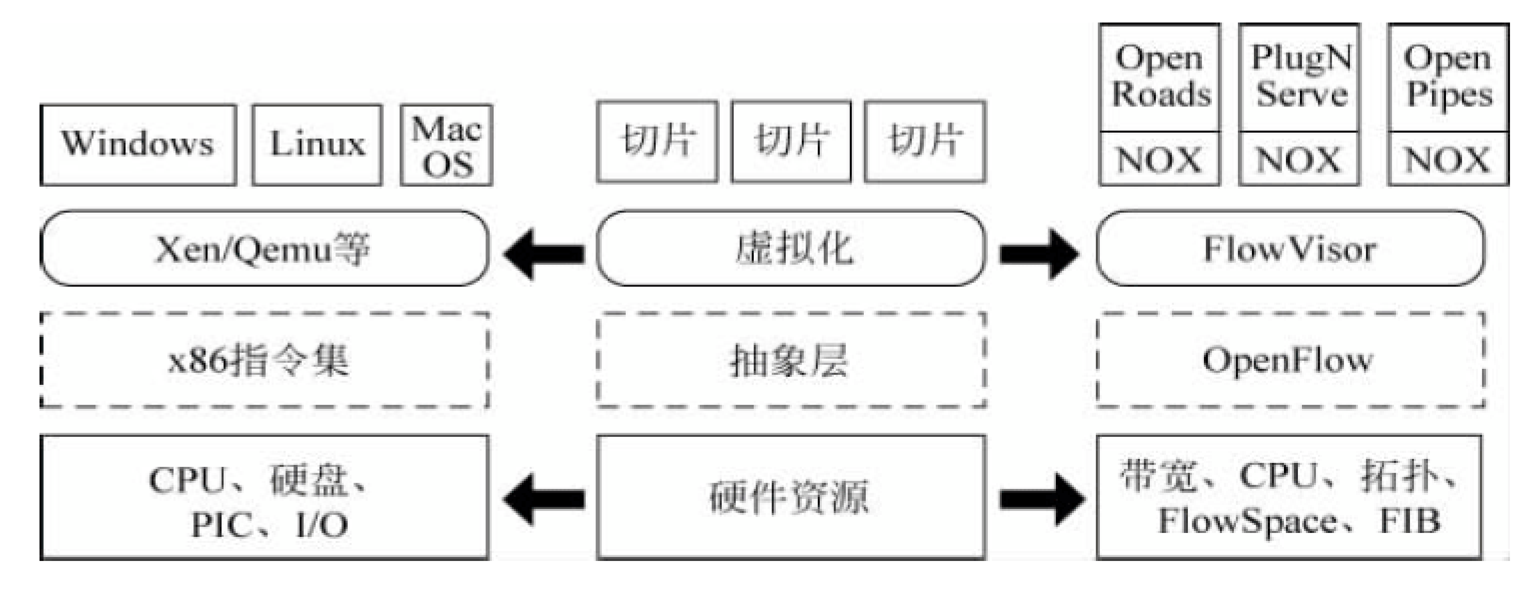


图 9-14 FlowVisor 和虚拟主机对比

FlowVisor 将网络中的所有流量看成全网可以调度的资源,管理员通过定义流空间(Flowspace)的方式来定义各个虚拟网络,而不是传统上通对路由器和交换机配置的方

式来实现网络虚拟化。

对于 OpenFlow 交换机来说，FlowVisor 也可以看成一个特殊的 OpenFlow 控制器，部署在真正的 OpenFlow 控制器和交换机之间，作为二者的透明代理。从 OpenFlow 交换机的视角看，FlowVisor 是一个网络控制器；从 OpenFlow 控制器的视角，FlowVisor 是一个物理的 OpenFlow 交换机，OpenFlow 交换机产生的信令首先发给 FlowVisor，FlowVisor 根据虚拟网络的配置将信令转发给相应的控制器。控制器产生信令下发给 FlowVisor，FlowVisor 对信令进行修改和约束后再转发给相应的 OpenFlow 交换机。可见 FlowVisor 能够根据虚拟网络流空间的定义规则进行消息的劫持、修改、分发等操作，而真正 OpenFlow 控制器只能看到 FlowVisor 向它提供的虚拟网络视图，这个视图包括虚拟网络的拓扑以及被分配给这个虚拟网络的流量，并不知道这些网络拓扑结构是 FlowVisor 虚拟出来的，流量是经过 FlowVisor 进行过滤的。

FlowVisor 使用 OpenFlow 协议控制底层的物理网络设备。可以将物理资源切分成很多份，将每一份切片作为一个虚拟网络视图提供给控制器。基于 FlowVisor 构建虚拟化网络时，可以指定另一个 FlowVisor 为网络切片的控制器，从而实现嵌套网络虚拟化，如图 9-15 所示。

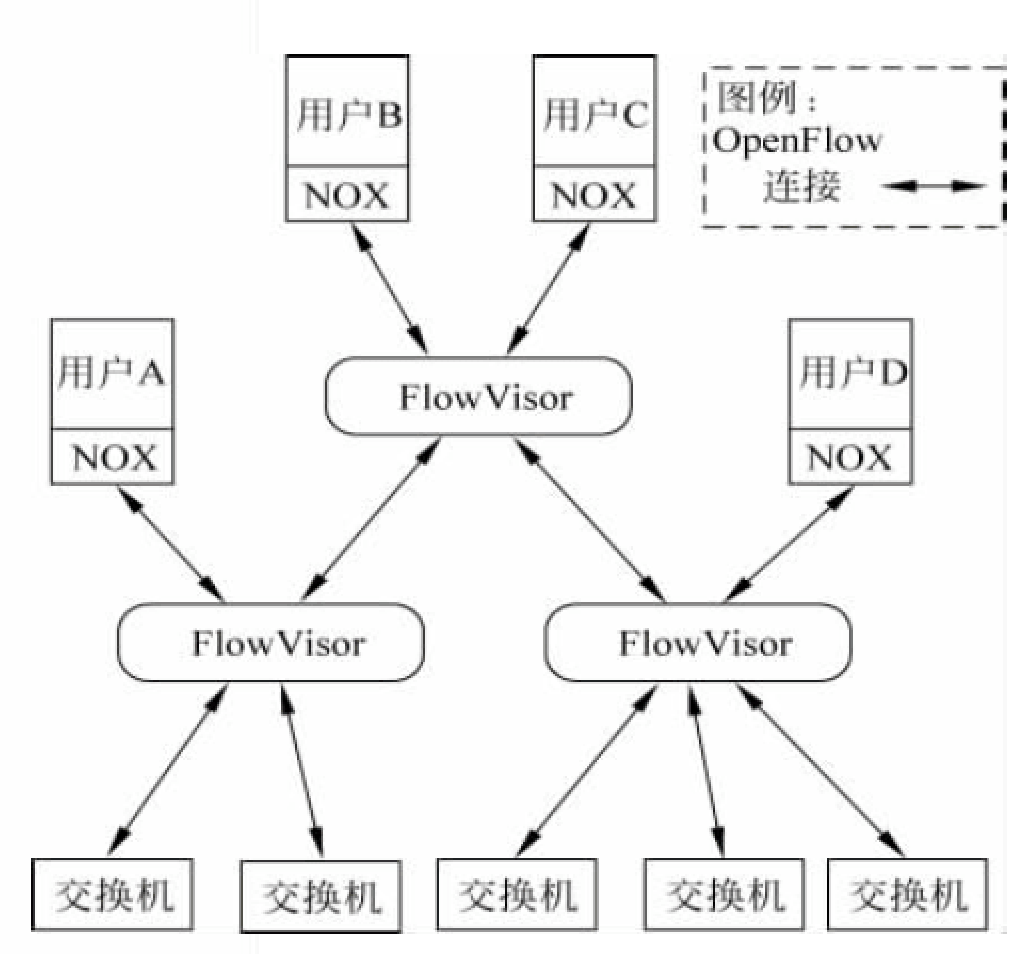


图 9-15 FlowVisor 组网示意图

FlowVisor 在设计过程中考虑了对如下几种资源的虚拟化，但 FlowVisor 不保证以下资源的虚拟化能够完全部署，在特定的条件下部分资源的虚拟化是无法实现的。

（1）带宽：每个虚拟网络所占用的带宽是可以配置的，这就需要对链路带宽进行抽象和分配。

（2）拓扑：每个虚拟网络拥有属于自己的网络节点及链路连接视图。控制器可以通过 OpenFlow 协议学习虚拟网络的拓扑，FlowVisor 能够为不同的虚拟网络控制器提供不同的虚拟网络拓扑。

（3）流量：不同虚拟网络的流量应该是互不干扰的，这就需要对流量进行抽象并管理，确保不同虚拟网络的流量能够严格地互相隔离。

(4) CPU 资源：网络设备用来处理数据平面的流量转发需要计算资源的开销，因此网络设备计算资源也需要被划分，交换机对控制信令的数据平面分组进行转发的过程都需要 CPU 的开销，如果没有适当的 CPU 划分，一个虚拟网络就可能影响其他虚拟网络的正常工作。

(5) 流表：交换机中的流表只有一个，需要多个虚拟网络共享，如果没有流表隔离机制，一个虚拟网络所下发的控制指令可能会影响到另一个虚拟网络的数据流量。

FlowVisor 中每一个虚拟网络是一个网络切片，每一个切片包含多条流空间(flowspace)。FlowVisor 中 flowspace 是定义在一台交换机之上的流量集合，它描述了一个切片所能够控制的流量特征。flowspace 的匹配范围涵盖了物理层、数据链路层、网络层和传输层的协议字段，这些字段如下：

In_port	Dl_src	Dl_dst	Dl_vlan	Dl_type	Nw_src	Nw_dst	Nw_proto	Nw_tos	Tp_src	Tp_dst

与 OpenFlow 1.0 定义的 12 元组不同，FlowVisor 的 FlowSpace 只有 11 元组，缺少了 VLAN 优先级字段，这是因为 FlowVisor 使用 VLAN 优先级字段的值提供基于差分服务的 QoS。FlowVisor 通过定义数据分组的 11 个包头字段确定一类网络流量的集合，如果这 11 个字段中某个字段没有定义，那么这条 flowspace 匹配这个字段的所有值。通过定义一个切片所包含的 flowspace，可以达到限定当前切片内数据流量类型的目的。一般来讲，对于某个特殊的应用，可以通过指定源/目的 MAC 地址、源/目的 IP 地址或者传输层的端口信息来把数据分组划分到一个切片内。

另外，每个 flowspace 都定义了一个控制器具有的行为权限限定。一个切片中的 flowspace 可以由一组文本配置文件定义三种权限。文本配置文件包含控制各种网络活动的规则，例如允许、只读和丢弃，其范围包括流量的来源 IP 地址、端口号或者数据包表头信息。

(1) 允许：虚拟网络控制器能够接收匹配这条 flowspace 的流表，并且虚拟网络控制器能够向交换机中添加控制这条 flowspace 的流表，即虚拟网络对这条 flowspace 定义的流量是可读可写的。

(2) 只读：匹配这条 flowspace 的流仅能被此虚拟网络的控制器接收监控，控制器对这条 flowspace 的控制操作都是非法的。

(3) 丢弃：匹配这条 flowspace 的流将被 FlowVisor 直接丢弃，虚拟网络的控制器也无法操作这条流。

FlowVisor 根据一个切片的 flowspace 推断出切片的拓扑。如果在一台交换机上定义了一个 flowspace，那么虚拟网络的拓扑就包含了这台交换机；如果这条 flowspace 的 in_port 是某个指定的端口，那么虚拟网络的拓扑只包含这个端口；如果一个 flowspace 的 in_port 没有指定端口，那么这个虚拟网络将包含这台交换机上的所有端口。

2. FlowVisor 工作流程

FlowVisor 主要由 FVClassifier、FVSlicer 和 FlowSpace 数据库三部分组成。FVClassifier 用于维护与物理 OpenFlow 交换设备的连接，处理 I/O 请求并记录每个物

理设备的端口、性能等信息，每个 FVClassifier 对应一个 OpenFlow 交换设备。FVSlicer 用于维护与控制器的连接，管理 OpenFlow 会话并对控制器下发的信令进行处理。当流从一个物理的 OpenFlow 交换设备到达 FlowSpace 后，FlowSpace 根据数据库中切片规则，把 OF 消息交给本切片内的 FVSlicer 发送给连接的控制器。[3]

FlowVisor 的工作流程如图 9-16 所示，控制器和交换机与 FlowVisor 之间产生的所有事件都进入到一个消息队列，通过 Poll Loop 轮询来处理。交换机交给控制器的数据分组通过 OFSwitchAcceptor 模块接收后，递交给对应的 FVClassifier 模块以对数据分组类型以及参数特征进行分析；随后此数据分组进入 FVSlicer 模块，此模块与数据库交互，实现数据分组与 FlowSpace 数据库的内容进行比对；最终由 FVSlicer 模块决定此数据分组所属的虚拟网络，并交给相应的 FVSlicer 转发给控制器。同理，控制器下发给交换机的数据分组会通过相反的过程送达交换机，但是在下发到物理交换机之前，FVSlicer 会对下行的信令进行检查，防止下行信令操作虚拟网络 FlowSpace 未定义的数据流。

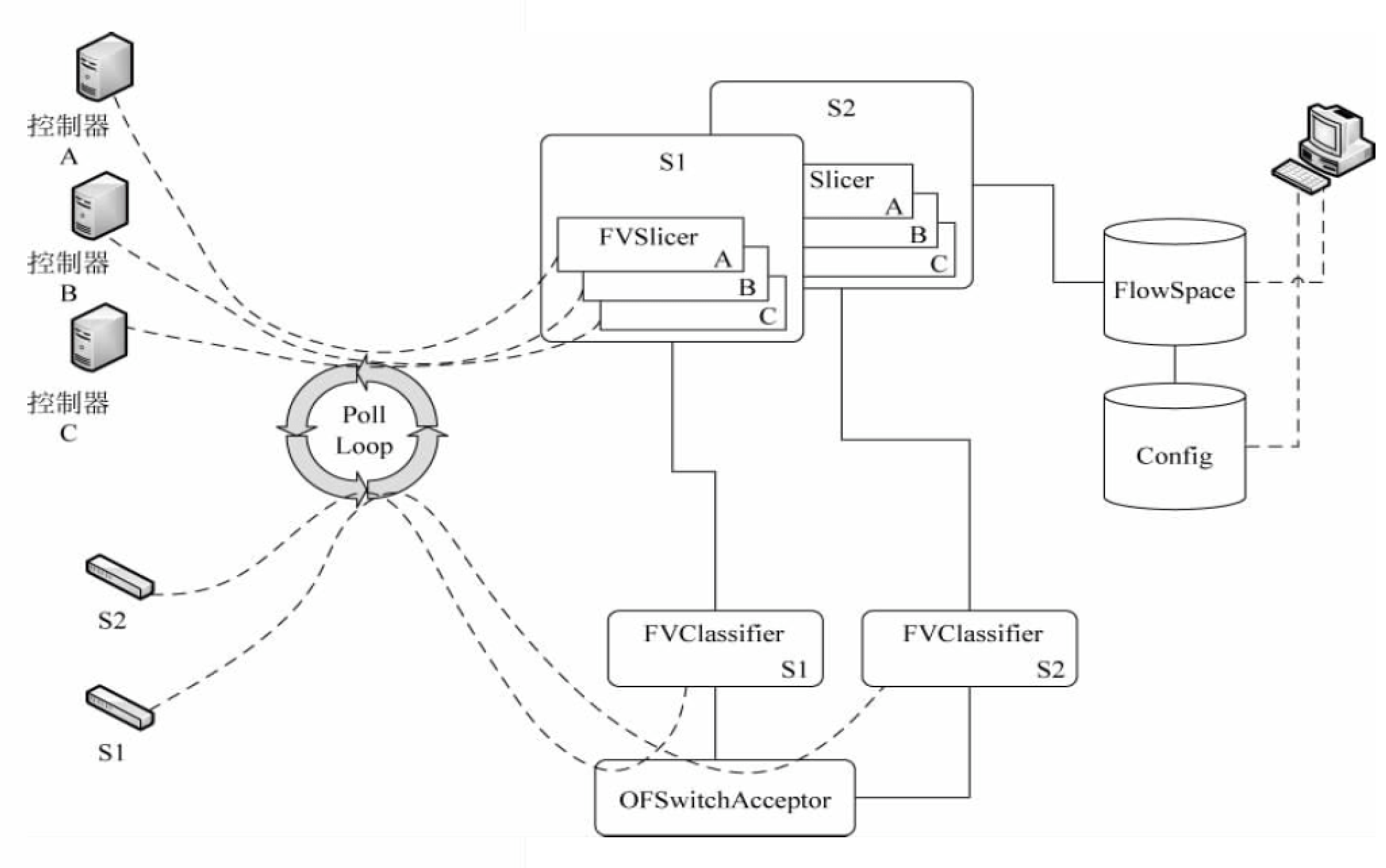


图 9-16 FlowVisor 工作原理

9.3.2 FlowVisor 安装使用

1. 安装

1）环境搭建

（1）安装 JDK1.7：

```
sudo apt-get install openjdk-7-jdk
```

（2）安装 ant：

```
sudo apt－get install Ant
```

2）获取源码

下载 flowvisor-1.0-MAINT.zip：

```
Git clone git://github.com/OPENWORKINGLAB/Flowvisor.git
```

3）make 操作进行编译和安装

```
cd /flowvisor
sudo make
```

编译结果如图 9-17 所示。

```
ubuntu@ubuntu:~$ cd flowvisor/
ubuntu@ubuntu:~/flowvisor$ sudo make
ant
Buildfile: /home/ubuntu/flowvisor/build.xml

init:

compile:
    [javac] /home/ubuntu/flowvisor/build.xml:60: warning: 'includeantruntime' wa
s not set, defaulting to build.sysclasspath=last; set to false for repeatable bu
ilds

dist:

BUILD SUCCESSFUL
Total time: 1 second
ubuntu@ubuntu:~/flowvisor$
```

图 9-17　编译结果

看到 BUILD SUCCESSFUL 的提示说明安装成功，如图 9-18 所示。

```
sudo make install
```

```
ubuntu@ubuntu:~/flowvisor$ sudo make install
ant
Buildfile: /home/ubuntu/flowvisor/build.xml

init:

compile:
    [javac] /home/ubuntu/flowvisor/build.xml:60: warning: 'includeantruntime' wa
s not set, defaulting to build.sysclasspath=last; set to false for repeatable bu
ilds

dist:

BUILD SUCCESSFUL
Total time: 0 seconds
./scripts/install-script.sh
Using source dir: ./scripts/..
Installation prefix (/usr/local): /usr/local
FlowVisor User (needs to already exist) (flowvisor): ubuntu
FlowVisor Group (needs to already exist) (flowvisor): ubuntu
```

图 9-18　安装结果

在输入完 User 和 Group 后，可以选择不输入密码，以免以后每次操作的时候都要输入密码。

4）启动 FlowVisor

在终端中输入命令：

```
flowvisor /etc/flowvisor/config.json
```

启动后界面如图 9-19 所示。

```
ubuntu@ubuntu:~$  flowvisor /etc/flowvisor/config.json
Starting FlowVisor
--- Setting logging level to NOTE
2016-05-25 19:38:58.741:INFO::Logging to StdErrLog::DEBUG=false via org.eclipse.
jetty.util.log.StdErrLog
2016-05-25 19:38:58.798:INFO::jetty-7.0.2.v20100331
2016-05-25 19:38:59.517:INFO::Started SslSelectChannelConnector@0.0.0.0:8181
```

图 9-19　启动 FlowVisor 界面

在这里要注意的是，FlowVisor 的服务端口是 8181。

2. 切片操作

1）创建切片

创建切片的命令格式：

```
fvctl add-slice <slice name> tcp:<controller IP address>:<controller port> <your e-mail>
```

具体命令：

```
fvctl -p 8181 add-slice s1 tcp:192.168.5.64:6633 wyf@bupt.com
```

创建成功如图 9-20 所示。

```
ubuntu@ubuntu:~$ fvctl -p 8181 add-slice slice1 tcp:192.168.226.129:6633 wyf@bupt.com
Password:
Slice password:
Slice slice1 was successfully created
ubuntu@ubuntu:~$
```

图 9-20　创建成功

要注意的是，TCP 条件后面的 IP 地址是控制器的 IP 地址。在这里使用的控制器是 OpenDaylight，端口就是 OpenDaylight 监听的端口。还有就是一个切片只能对应一个控制器。

下面来看看创建的切片。

2）查看所有的切片

命令格式：

```
fvctl -p <port> list-slices
```

具体命令：

```
fvctl -p 8181 list-slices
```

结果如图 9-21 所示。

```
ubuntu@ubuntu:~$ fvctl -p 8181 list-slices
Password:
Configured slices:
fvadmin          --> enabled
slice1           --> enabled
ubuntu@ubuntu:~$
```

图 9-21　查看所有切片

3）查看具体切片

命令格式：

fvctl -p <port> list－slice－info <slicename>

具体命令：

fvctl －p 8181 list－slice－info slice1

结果如图 9-22 所示。

```
ubuntu@ubuntu:~$ fvctl -p 8181 list-slice-info slice1
Password:
Note: No switches connected; no runtime stats available
{
  "admin-contact": "wyf@bupt.com",
  "admin-status": true,
  "controller-url": "tcp:192.168.226.129:6633",
  "drop-policy": "exact",
  "recv-lldp": false,
  "slice-name": "slice1"
}
```

图 9-22　查看具体切片

4）显示切片的统计信息

命令格式：

fvctl －p <port> list－slice－stats <slicename>

具体命令：

fvctl －p 8181 list－slice－stats slice1

结果如图 9-23 所示。

5）修改切片

命令格式：

fvctl －p <port> update－slice [option]<slice－name>

具体命令：

(1）停用切片。

fvctl －p 8181 update－slice －－disable－slice slice1

结果如图 9-24 所示。

```
ubuntu@ubuntu:~$ fvctl -p 8181 list-slice-stats slice1
Password:
{
  "drop": {
    "Total": {
      "ECHO_REQUEST": 8,
      "PORT_STATUS": 16
    },
    "slicer_slice1_dpid=00:00:00:00:00:00:00:03": {
      "ECHO_REQUEST": 2,
      "PORT_STATUS": 4
    },
    "slicer_slice1_dpid=00:00:00:00:00:00:00:04": {
      "ECHO_REQUEST": 2,
      "PORT_STATUS": 4
    }
  },
  "rx": {
    "Total": {}
  },
  "tx": {
    "Total": {}
  }
}
```

图 9-23　显示 Slice 的统计信息

```
ubuntu@ubuntu:~$ fvctl -p 8181 update-slice --disable-slice slice1
Password:
Slice slice1 has been successfully updated
```

图 9-24　停用切片

（2）启用切片。

fvctl -p 8181 update-slice --enable-slice slice1

结果如图 9-25 所示。

```
ubuntu@ubuntu:~$ fvctl -p 8181 update-slice --enable-slice slice1
Password:
Slice slice1 has been successfully updated
```

图 9-25　启用切片

6）删除切片

命令格式：

fvctl -p <port> remove-slice <slicename>

具体命令：

fvctl -p 8181 remove-slice slice1

结果如图 9-26 所示。

```
ubuntu@ubuntu:~$ fvctl -p 8181 remove-slice slice1
Password:
Slice slice1 has been deleted
```

图 9-26　删除切片

3. 流规则操作

添加 flowspace 的命令格式：

fvctl add－flowspace [options] <flowspace－name> <dpid> <priority> <match> <slice－perm>

流表中常用字段的名称和含义如表 9-2 所示。

表 9-2 FlowVisor 流表中常用字段

字段名	含义
priority	优先级，0～65535
in_port	进入端口
dl_vlan	VLAN ID，0xffff 表示匹配 VLAN 包，否则指定为 0～4095 表示 12 位的 VLAN ID
dl_vpcp	VLAN 优先级，最外层 VLAN 头的 PCP 域
dl_src	匹配源 MAC 地址
dl_dst	匹配源目的 MAC 地址
dl_type	匹配以太网协议类型，采用 0～65535 表示
nw_src	源 IP 地址
nw_dst	目的 IP 地址
nw_proto	IP 协议类型，十进制数 0～255
nw_tos	IP Tos 位，采用 0～255 表示
tp_src	TCP/UDP 源端口
tp_dst	TCP/UDP 目的端口
wildcards	匹配规则
actions	切片行为，指的是切片对这个 flowspace 拥有的权限，DELEGATE＝1，READ＝2，WRITE＝4。actions 的值为这三个的组合，所有取值范围为 1～7

下面介绍具体指令。

1）基于 IP 地址

（1）fvctl -p 8181 add-flowspace flowspace1 all 100 nw_src＝10.0.0.3，nw_dst＝10.0.0.1 slice1＝7。结果如图 9-27 所示。

```
ubuntu@ubuntu:~$ fvctl -p 8181 add-flowspace flowspace1 all 100 nw_src=10.0.0.3,
nw_dst=10.0.0.1 slice1=7
Password:
FlowSpace flowspace1 was added with request id 1.
```

图 9-27 基于 IP 地址(1)

（2）fvctl -p 8181 add-flowspace flowspace1 all 100 nw_src＝10.0.0.1，nw_dst＝10.0.0.3 slice1＝7。结果如图 9-28 所示。

```
ubuntu@ubuntu:~$ fvctl -p 8181 add-flowspace flowspace1 all 100 nw_src=10.0.0.1,
nw_dst=10.0.0.3 slice1=7
Password:
FlowSpace flowspace1 was added with request id 2.
```

图 9-28 基于 IP 地址(2)

2）基于 MAC 地址

在 Mininet 的 CLI 命令窗口中查看主机 h1 和 h3 的 MAC 地址分别为 da:0d:c9:37:d4:51 和 9a:4d:96:96:f4:47。因此相应指令如下：

（1）fvctl -p 8181 add-flowspace flowspace2 all 100 dl_src=9a:4d:96:96:f4:47,dl_dst=da:0d:c9:37:d4:51 slice1=7。结果如图 9-29 所示。

```
ubuntu@ubuntu:~$ fvctl -p 8181 add-flowspace flowspace2 all 100 dl_src=9a:4d:96:
96:f4:47,dl_dst=da:0d:c9:37:d4:51 slice1=7
Password:
FlowSpace flowspace2 was added with request id 3.
```

图 9-29　基于 MAC 地址(1)

（2）fvctl -p 8181 add-flowspace flowspace2 all 100 dl_src=da:0d:c9:37:d4:51,dl_dst=9a:4d:96:96:f4:47 slice1=7。结果如图 9-30 所示。

```
ubuntu@ubuntu:~$ fvctl -p 8181 add-flowspace flowspace2 all 100 dl_src=da:0d:c9:
37:d4:51,dl_dst=9a:4d:96:96:f4:47 slice1=7
Password:
FlowSpace flowspace2 was added with request id 4.
```

图 9-30　基于 MAC 地址(2)

3）基于端口

（1）fvctl -p 8181 add-flowspace flowspace3 all 100 in_port=1 slice1=7。结果如图 9-31 所示。

```
ubuntu@ubuntu:~$ fvctl -p 8181 add-flowspace flowspace3 all 100 in_port=1 slice1
=7
Password:
FlowSpace flowspace3 was added with request id 5.
```

图 9-31　基于端口(1)

（2）fvctl -p 8181 add-flowspace flowspace3 all 100 in_port=3 slice1=7。结果如图 9-32 所示。

```
ubuntu@ubuntu:~$ fvctl -p 8181 add-flowspace flowspace3 all 100 in_port=3 slice1
=7
Password:
FlowSpace flowspace3 was added with request id 6.
```

图 9-32　基于端口(2)

9.3.3　FlowVisor 应用实例

下面介绍 FlowVisor 是如何在 Mininet 和 OpenDaylight 之间工作的。首先启动 FlowVisor，之后操作 Mininet 部分。

1. Mininet 部分

1）创建虚拟拓扑

使用以下指令创建拓扑连接到 FlowVisor：

```
sudo mn --topo=tree,3 --controller=remote,ip=192.168.48.128,port=6464
```

在此通过以上指令创建了一个拥有三层树形拓扑结构的网络,--controller＝remote表示网络使用远程控制器 remote; ip 为 FlowVisor 所在主机的 ip; port 为 FlowVisor 的监听端口,这里将其修改为 6464。创建后的效果如图 9-33 所示。

```
ubuntu@ubuntu:~$ sudo mn --topo=tree,3 --controller=remote,ip=192.168.48.128,por
t=6464
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(h1, s3) (h2, s3) (h3, s4) (h4, s4) (h5, s6) (h6, s6) (h7, s7) (h8, s7) (s1, s2)
 (s1, s5) (s2, s3) (s2, s4) (s5, s6) (s5, s7)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7
*** Starting CLI:
mininet>
```

图 9-33 创建效果

2) 查看 FlowVisor 主机的 net

当虚拟网络创建好之后,就可以看到 FlowVisor 的监听端口建立的 TCP 连接。通过如下指令查看:

```
netstat -an | grep 6464
```

结果如图 9-34 所示。

```
ubuntu@ubuntu:~$ netstat -an | grep 6464
tcp        0      0 0.0.0.0:6464            0.0.0.0:*               LISTEN
tcp        0      0 192.168.48.128:50732    192.168.48.128:6464     ESTABLISHED
tcp        0      0 192.168.48.128:50729    192.168.48.128:6464     ESTABLISHED
tcp        0      0 192.168.48.128:50731    192.168.48.128:6464     ESTABLISHED
tcp        0      0 192.168.48.128:6464     192.168.48.128:50730    ESTABLISHED
tcp        0      0 192.168.48.128:6464     192.168.48.128:50733    ESTABLISHED
tcp        0      0 192.168.48.128:50733    192.168.48.128:6464     ESTABLISHED
tcp        0      0 192.168.48.128:6464     192.168.48.128:50728    ESTABLISHED
tcp        0      0 192.168.48.128:50728    192.168.48.128:6464     ESTABLISHED
tcp        0      0 192.168.48.128:50727    192.168.48.128:6464     ESTABLISHED
tcp        0      0 192.168.48.128:6464     192.168.48.128:50729    ESTABLISHED
tcp        0      0 192.168.48.128:6464     192.168.48.128:50727    ESTABLISHED
tcp        0      0 192.168.48.128:6464     192.168.48.128:50732    ESTABLISHED
tcp        0      0 192.168.48.128:6464     192.168.48.128:50731    ESTABLISHED
tcp        0      0 192.168.48.128:50730    192.168.48.128:6464     ESTABLISHED
ubuntu@ubuntu:~$
```

图 9-34 查看 FlowVisor 主机的 net

从中可以看到创建了 7 个交换机,这 7 个交换机和 FlowVisor 建立的 TCP 连接。

3) 查看 FlowVisor 中的 datapaths

使用 datapaths 指令可以查看连接到 FlowVisor 的交换机:

```
fvctl -p 8181 list-datapaths
```

结果如图 9-35 所示。

```
ubuntu@ubuntu:~$ fvctl -p 8181 list-datapaths
Password:
Connected switches:
  1 : 00:00:00:00:00:00:00:01
  2 : 00:00:00:00:00:00:00:02
  3 : 00:00:00:00:00:00:00:03
  4 : 00:00:00:00:00:00:00:04
  5 : 00:00:00:00:00:00:00:05
  6 : 00:00:00:00:00:00:00:06
  7 : 00:00:00:00:00:00:00:07
ubuntu@ubuntu:~$
```

图 9-35 查看 FlowVisor 中的 datapaths

2. FlowVisor 部分

1）创建切片

按照 9.3.2 节中的步骤启动 FlowVisor 并创建切片 slice1。

2）创建规则

在不创建规则的情况下，OpenDaylight 是无法和 FlowVisor 建立连接的。即使创建了 slice，并且提示是成功的，也无法连接。创建全网络连通的流表项：

```
fvctl -p 8181 add-flowspace fs1 all 100 any s1=7
```

创建完，终端不会显示什么有效信息，下面看一下 OpenDaylight 部分。

3. OpenDaylight 部分

1）控制端显示

到这里就创建好了一个网络虚拟化实例，控制端显示的信息，如图 9-36 所示。

```
2016-06-02 03:06:44.538 PDT [http-bio-8080-exec-8] INFO  o.o.c.u.internal.UserMa
nager - Local Authentication Succeeded for User: "admin"
2016-06-02 03:06:44.541 PDT [http-bio-8080-exec-8] INFO  o.o.c.u.internal.UserMa
nager - User "admin" authorized for the following role(s): [Network-Admin]
2016-06-02 03:07:08.991 PDT [ControllerI/O Thread] INFO  o.o.c.p.o.core.internal
.Controller - Switch:192.168.48.128:48355 is connected to the Controller
2016-06-02 03:07:12.700 PDT [ControllerI/O Thread] INFO  o.o.c.p.o.core.internal
.Controller - Switch:192.168.48.128:48356 is connected to the Controller
2016-06-02 03:07:13.634 PDT [ControllerI/O Thread] INFO  o.o.c.p.o.core.internal
.Controller - Switch:192.168.48.128:48357 is connected to the Controller
2016-06-02 03:07:13.875 PDT [ControllerI/O Thread] INFO  o.o.c.p.o.core.internal
.Controller - Switch:192.168.48.128:48358 is connected to the Controller
2016-06-02 03:07:14.712 PDT [ControllerI/O Thread] INFO  o.o.c.p.o.core.internal
.Controller - Switch:192.168.48.128:48359 is connected to the Controller
2016-06-02 03:07:15.020 PDT [ControllerI/O Thread] INFO  o.o.c.p.o.core.internal
.Controller - Switch:192.168.48.128:48360 is connected to the Controller
2016-06-02 03:07:15.060 PDT [ControllerI/O Thread] INFO  o.o.c.p.o.core.internal
.Controller - Switch:192.168.48.128:48361 is connected to the Controller
```

图 9-36 控制端显示的信息

这里能看到 7 个虚拟交换机和控制器通过 TCP 相连。

2）Web 端显示

如图 9-37 所示是 Web 端的拓扑结构，当然这些交换机并不是真的存在，是通过 Mininet 虚拟出来的拓扑结构。

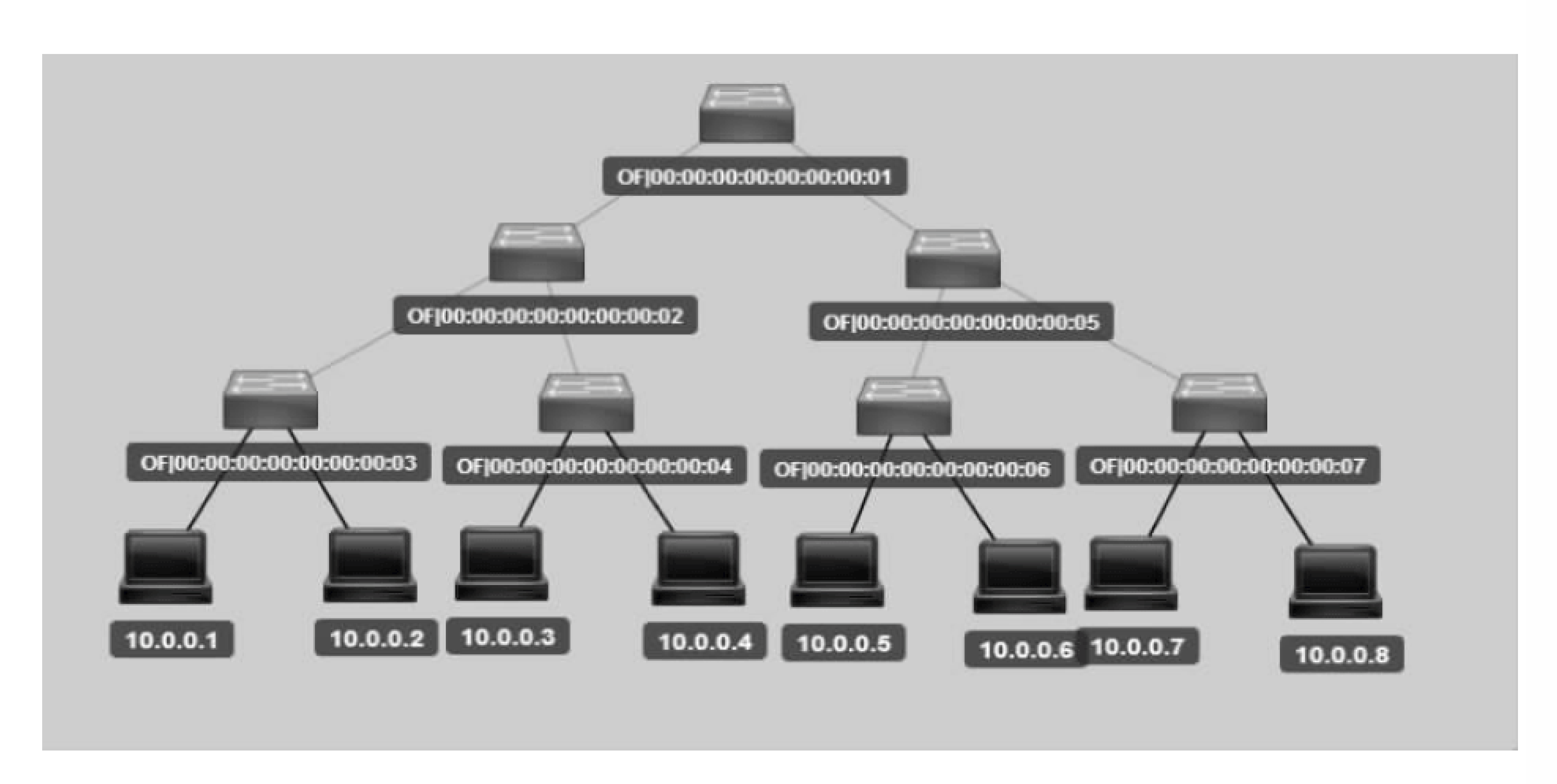


图 9-37　Web 端的拓扑结构

并且可以查看到创建的交换机的节点信息如图 9-38 所示。

Nodes Learned

Search

Node Name	Node ID	Tier Name	MAC Address	Ports
None	OF|00:00:00:00:00:00:00:02	Unknown (Tier-0)	00:00:00:00:00:02	3
None	OF|00:00:00:00:00:00:00:03	Access (Tier-1)	00:00:00:00:00:03	3
None	OF|00:00:00:00:00:00:00:01	Unknown (Tier-0)	00:00:00:00:00:01	2
None	OF|00:00:00:00:00:00:00:04	Access (Tier-1)	00:00:00:00:00:04	3
None	OF|00:00:00:00:00:00:00:07	Access (Tier-1)	00:00:00:00:00:07	3
None	OF|00:00:00:00:00:00:00:06	Access (Tier-1)	00:00:00:00:00:06	3
None	OF|00:00:00:00:00:00:00:05	Unknown (Tier-0)	00:00:00:00:00:05	3

1-7 of 7 items　Page 1 of 1

图 9-38　交换机的节点信息

9.3.4　基于 FlowVisor 的虚网划分

1. 实验原理

本实验基于 FlowVisor 进行虚网划分，验证同一网络的不同虚网下面的主机之间在逻辑上不连通是不能互相通信的，而相同虚网下的主机之间可以互相通信。

2. 实验步骤

1）启动 FlowVisor

按照上面介绍的步骤启动 FlowVisor。

2）创建拓扑

```
sudo mn --topo=single,4 --controller=remote,ip=192.168.48.129,port=6464
```

结果如图 9-39 所示。

```
ubuntu@ubuntu:~$ sudo mn --topo=single,4 --controller=remote,ip=192.168.48.129,p
ort=6464
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
*** Starting 1 switches
s1
*** Starting CLI:
mininet>
```

图 9-39　创建拓扑

从执行的指令或创建的拓扑结构可以看出，在此创建了有 4 个端口的交换机，每个端口下连接一个主机，这是为下面的虚网划分、下发静态流表做准备。

3）创建切片

这里要创建的是两个切片，分别是 s1 和 s2。由于一台虚拟机上只能创建一个切片，因此在此启动两台虚拟机，分别运行 ODL 控制器，并创建切片。

两台控制器的 IP 和监听端口分别如下：

```
ip = 192.168.48.129,port = 6633
ip = 192.168.48.130,port = 6633
```

创建结果如图 9-40 所示。

```
ubuntu@ubuntu:~$ fvctl -p 8181 add-slice slice1 tcp:192.168.48.129:6633 wyf@bupt
.com
Password:
Slice password:
Slice slice1 was successfully created
ubuntu@ubuntu:~$ fvctl -p 8181 add-slice slice2 tcp:192.168.48.130:6633 wyf@bupt
.com
Password:
Slice password:
Slice slice2 was successfully created
```

图 9-40　创建切片

4）创建流规则

在这里需要创建两条流规则，分别为切片 1 和切片 2 划分主机。对于同一个切片的流规则其流规则的名字不可以重复，但是对于不同切片的流规则其名字则可以重复。

为切片 1 创建的流规则如下：

```
fvctl  - p 8181 add - flowspace flowspace1 all 100 nw_src = 10.0.0.1,nw_dst = 10.0.0.3 slice1 = 7
fvctl  - p 8181 add - flowspace flowspace1 all 100 nw_src = 10.0.0.3,nw_dst = 10.0.0.1 slice1 = 7
```

为切片 2 创建的流规则如下：

```
fvctl -p 8181 add-flowspace flowspace1 all 100 nw_src=10.0.0.2,nw_dst=10.0.0.4 slice2=7
fvctl -p 8181 add-flowspace flowspace1 all 100 nw_src=10.0.0.4,nw_dst=10.0.0.2 slice2=7
```

5）查看流规则

```
fvctl -p 8181 list-flowspace
```

结果如图 9-41 所示。


```
{"force-enqueue": -1, "name": "flowspace1", "slice-action": [{"slice-name": "slice1", "permi
ssion": 7}], "queues": [], "priority": 100, "dpid": "all_dpids", "id": 2, "match": {"wildcar
ds": 3145983, "nw_src": "10.0.0.1", "nw_dst": "10.0.0.3"}}
{"force-enqueue": -1, "name": "flowspace1", "slice-action": [{"slice-name": "slice1", "permi
ssion": 7}], "queues": [], "priority": 100, "dpid": "all_dpids", "id": 3, "match": {"wildcar
ds": 3145983, "nw_src": "10.0.0.3", "nw_dst": "10.0.0.1"}}
{"force-enqueue": -1, "name": "flowspace1", "slice-action": [{"slice-name": "slice2", "permi
ssion": 7}], "queues": [], "priority": 100, "dpid": "all_dpids", "id": 4, "match": {"wildcar
ds": 3145983, "nw_src": "10.0.0.2", "nw_dst": "10.0.0.4"}}
{"force-enqueue": -1, "name": "flowspace1", "slice-action": [{"slice-name": "slice2", "permi
ssion": 7}], "queues": [], "priority": 100, "dpid": "all_dpids", "id": 5, "match": {"wildcar
ds": 3145983, "nw_src": "10.0.0.4", "nw_dst": "10.0.0.2"}}
```


图 9-41　查看流规则

6）查看创建的交换机和主机

下面查看创建的主机交换机的信息。查看信息是在 ODL 的 Web 端，如图 9-42 所示。

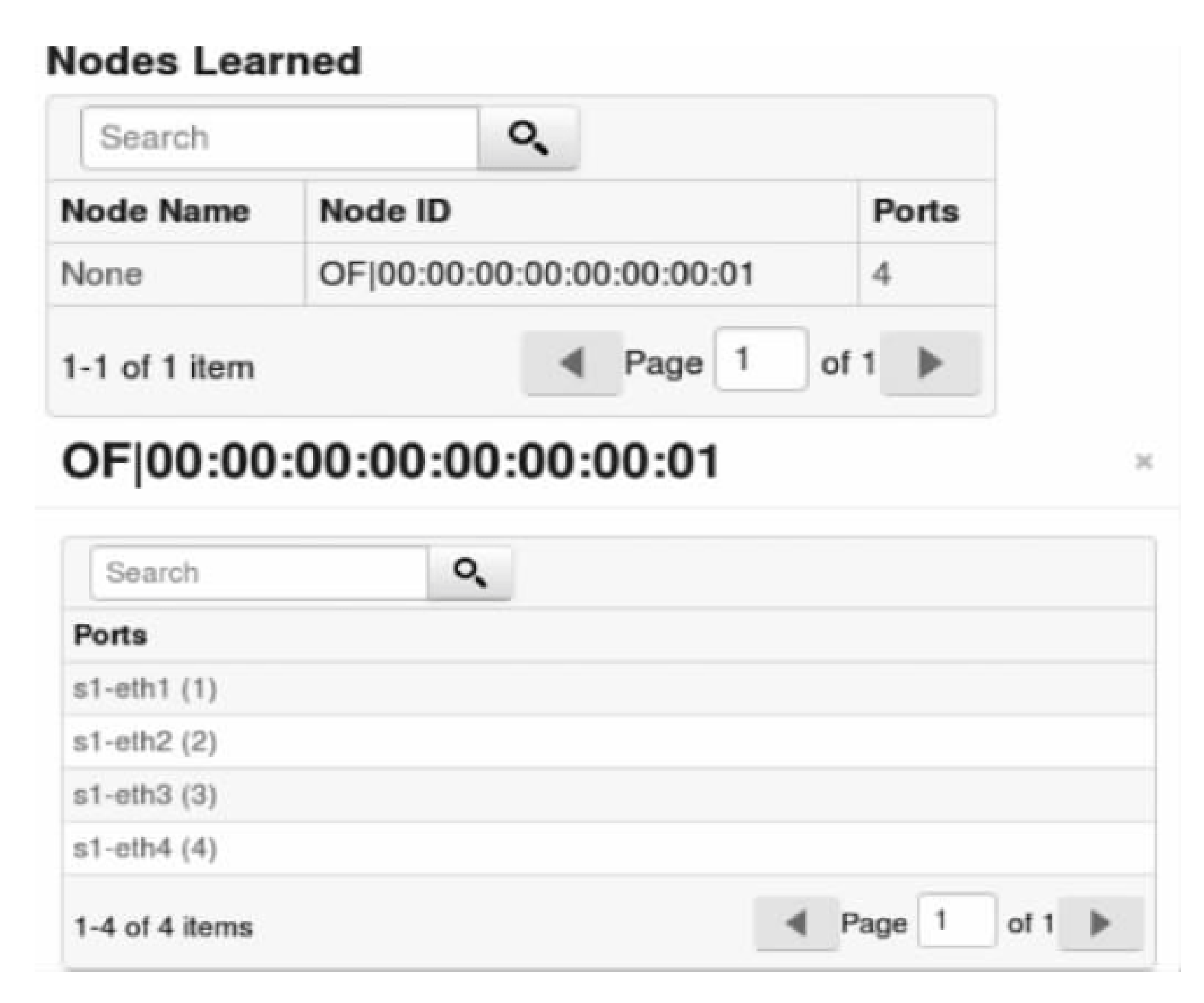


图 9-42　查看创建的交换机和主机

在这里可以验证在之前创建的是一个交换机连接着 4 个主机，这 4 个主机分别连接在 s1-eth1、s1-eth2、s1-eth3、s1-eth4 的 4 个端口中。

7）虚网划分后进行 ping 操作

下面看看主机的连通情况，在这里在 Mininet 中进行 pingall 操作。

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3 X
h2 -> X X h4
h3 -> h1 X X
h4 -> X h2 X
*** Results: 66% dropped (8/12 lost)
```

之后分别在两台虚拟机的 ODL 的 Web 端查看网络的拓扑结构，可以看到在第一个虚网中只有 h1 和 h3 两台主机，在第二个虚网中只有 h2 和 h4 两台主机，实现了虚网的划分，如图 9-43 所示。

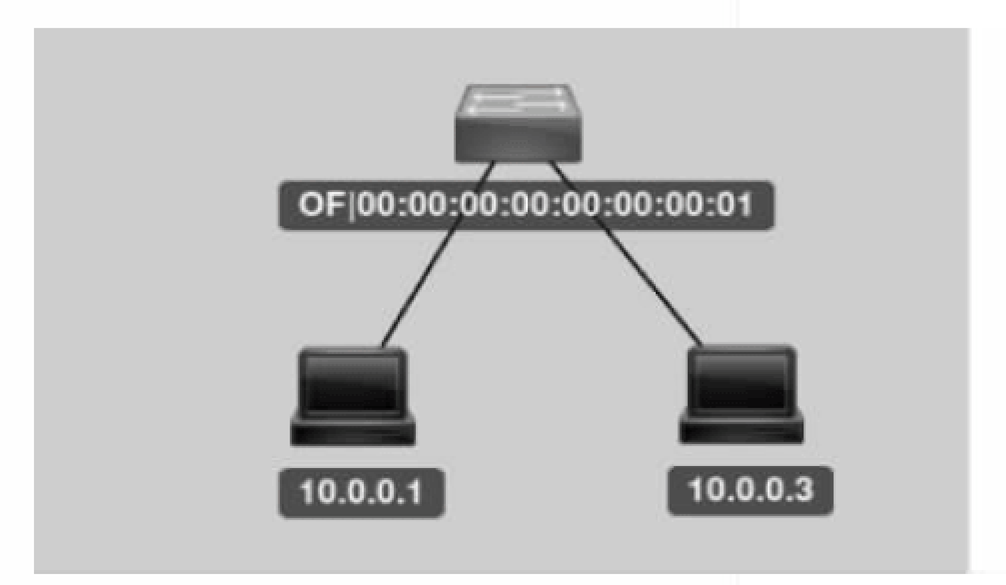




图 9-43　网络的拓扑结构

从图 9-43 中可以看出，h1 和 h3 是在同一个虚网 s1 中，h2 和 h4 在同一个虚网 s2 中，还可以看出，虚网划分后同一交换机上不同虚网内的主机之间是不能直接通信的。

3．实验结论

在虚网划分后，不同虚网间的主机无法直接通信，在逻辑上是不连通的。在创建流表时，如果创建的流表涉及的端口分别在两个虚网之间也是下发不成功的。

参考文献

[1]　http://blog.sina.com.cn/s/blog_8a01583f0101mns3.html
[2]　http://www.sdnap.com/sdnap-post/4370.html
[3]　http://www.cnblogs.com/zxqstrong/p/4785266.html

第10章 可编程异构网络管理平台

理论分析和仿真验证为异构网络下所提的各种策略提供了依据和基础。然而，由于异构网络实际实现比较复杂，目前大多数策略的验证依赖于程序仿真。设计一种结合实际的软硬件的物理平台，为研究异构网络中负载均衡、节点休眠等各种优化策略提供数据支持是十分有必要的。本章就如何构建可编程异构网络管理平台进行展开。

10.1 网络配置管理技术简介

网络管理即通过网络管理程序对网络上的资源进行集中化的管理操作，管理内容包括网络配置、网络性能、网络审计、网络问题以及网络变化等，以使网络有效地运行，为用户提供一定质量水平的服务。随着网络规模日益增加，网络设备越来越多，对网络配置方式的要求也越来越高。

SNMP(simple network management protocol)是一种著名的网络管理协议，它被广泛应用于数据网络领域。虽然 SNMP 可以通过 SET 操作对网络设备进行配置，但是却很少被用于网络配置，并且大多数设备都禁止了 SET 的功能。因为 SNMP 由于协议的简单性与数据建模语言的缺陷而无法对复杂的网络进行有效的配置和创建复杂的配置数据模型[1]。常用的配置管理方式还有 CLI(command line interface)命令方式，通过不同配置数据输入不同命令，配置起来也很烦琐且容易出错。不同厂商没有统一命令标准，也无法对网络中设备进行统一的配置，不利于管理员操作。

10.1.1 基于 NETCONF 的网络配置管理

为了改善配置管理方式的不足，IETF 标准化组织成立了 NETCONF 工作组，并在 2006 年通过了 RFC4741-4744，建立了新一代的网络管理协议 NETCONF(network configuration)，并且不断通过新的标准对其进行完善。

NETCONF 协议是一种新的基于 XML 的配置管理协议。它采

用 XML 作为配置数据和协议消息内容的数据编码方式，采用基于 TCP（传输控制协议）的传送机制，用 SSHvZ、SOAP、BEEP 等传输方式通过简单的 RPC（远程过程调用）方式来获取、更新或删除设备中相应的部分或所有管理信息[2]。将网管数据以标准的 XML 格式存储可以做到网管技术与 WWW 技术的真正融合。

XML 是指可扩展置标语言（extensible markup language），类似于 HTML。XML 设计宗旨是传输数据，而非显示数据，这一点与 HTML 不同。XML 的标签需要用户去预定义，XML 文档是一种树形结构，文档中必须包含根元素，该元素是其他元素的父元素。父元素拥有子元素，相同层级的元素称为同胞，所有元素均可以拥有文本内容和属性[3]。

NETCONF 协议分为 4 层：传输协议层、RPC 层、操作层与内容层，如图 10-1 所示。

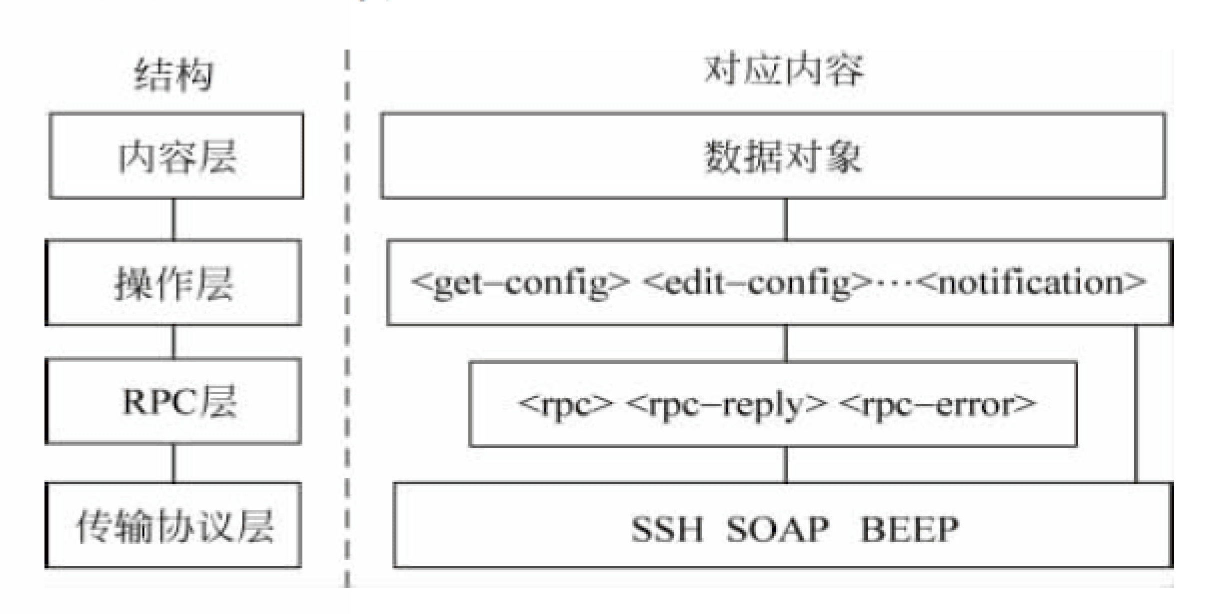


图 10-1 NETCONF 协议结构

NETCONF 协议结构中最内层是内容层，最外层是传输协议层。在构造 NETCONF 报文时，将从内到外层层封装，而解封报文时，则从外到内分析报文。协议的每层结构是相对独立的，可以单独地进行研究和实现。协议的传输协议层提供管理端和代理进行安全可靠通信的方案。为了保证配置管理的安全可靠性，弥补 SNMP 的安全缺陷，NETCONF 协议必须提供面向连接的安全传输机制，通过建立会话和关闭会话的方法建立连接。同时 NETCONF 协议为了提高协议的可扩展性，可以使用多种不同的传输机制。RFC4742-4744 分别定义了基于 SSH、SOAP、BEEP 的通信规范，这些安全协议通过加密和认证等方法来保证网络连接的安全性。实现 NETCONF 协议的时候可以同时实现多种传输机制，其中强制实现的是 SSH。

内容层表示的是被管对象的集合。内容层的内容需要来自数据模型中，而原有的 MIB 等数据模型对于配置管理存在着如不允许创建和删除行、对应的 MIB 不支持复杂的表结构等缺陷，因此内容层的内容没有定义在 RFC4741 中。到目前为止，NETCONF 内容层是唯一没有被标准化的层，没有标准的 NETCONF 数据建模语言和数据模型，其相关理论还在进一步讨论中。

操作层定义了一系列在 RPC 中应用的基本操作，这些操作将组成 NETCONF 的基本能力。为了简单的目的，与 SNMP 只定义的 5 种包含 get、set 的基本操作相比，NETCONF 全面地定义了取值操作、配置操作、锁操作和会话操作，其中 get、get-config 命令用来对设备进行取值操作；edit-config、copy-config、delete-config 命令则是用于配置设备参数；lock 和 unlock 则是在对设备进行操作时为防止并发产生混乱的锁行为；

close-session 和 kill-session 则是相对比较上层的操作，用于结束一个会话操作。

RPC 层为 RPC 模块的编码提供了一个简单的、传输协议无关的机制。通过使用 < rpc >和< rpc-reply >元素对 NETCONF 协议的客户端（网络管理者或网络配置应用程序）和服务器端（网络设备）的请求和响应数据（即操作层和内容层的内容）进行封装，正常情况下< rpc-reply >元素封装客户端所需的数据或配置成功的提示信息，当客户端请求报文存在错误或服务器端处理不成功时，服务器端在< rpc-reply >元素中会封装一个包含详细错误信息的< rpc- error >元素来反馈给客户端。

NETCONF 的使用使网络设备的信息查看与配置不用远程登录到设备上去，NETCONF 支持所有可以用 SHOW 命令查看的信息，包括配置信息与运行状态信息等。

10.1.2 大规模网络自动化部署技术

NETCONF 协议和相关工具的出现，虽然极大地方便了对网络的配置与管理，但是 NETCONF 大部分用于对单个设备进行操作，并不适合对网络中大规模的设备同时进行整体操作。

自动化部署技术的出现，使对网络中大规模设备同时操作成为可能。自动化部署又可以称为自动化配置，是指设备在启动时自动获取并执行配置文件。当网络规模较大而且分散时，网络管理员通过登录每一台设备进行配置的工作量很大，这时管理员可以考虑使用设备自动配置。只需将配置文件保存在指定的存储服务器上，设备启动时可以自动从文件存储服务器上获取配置文件并执行，简化了网络配置，降低了网络管理员的工作量。自动配置的典型网络环境如图 10-2 所示。

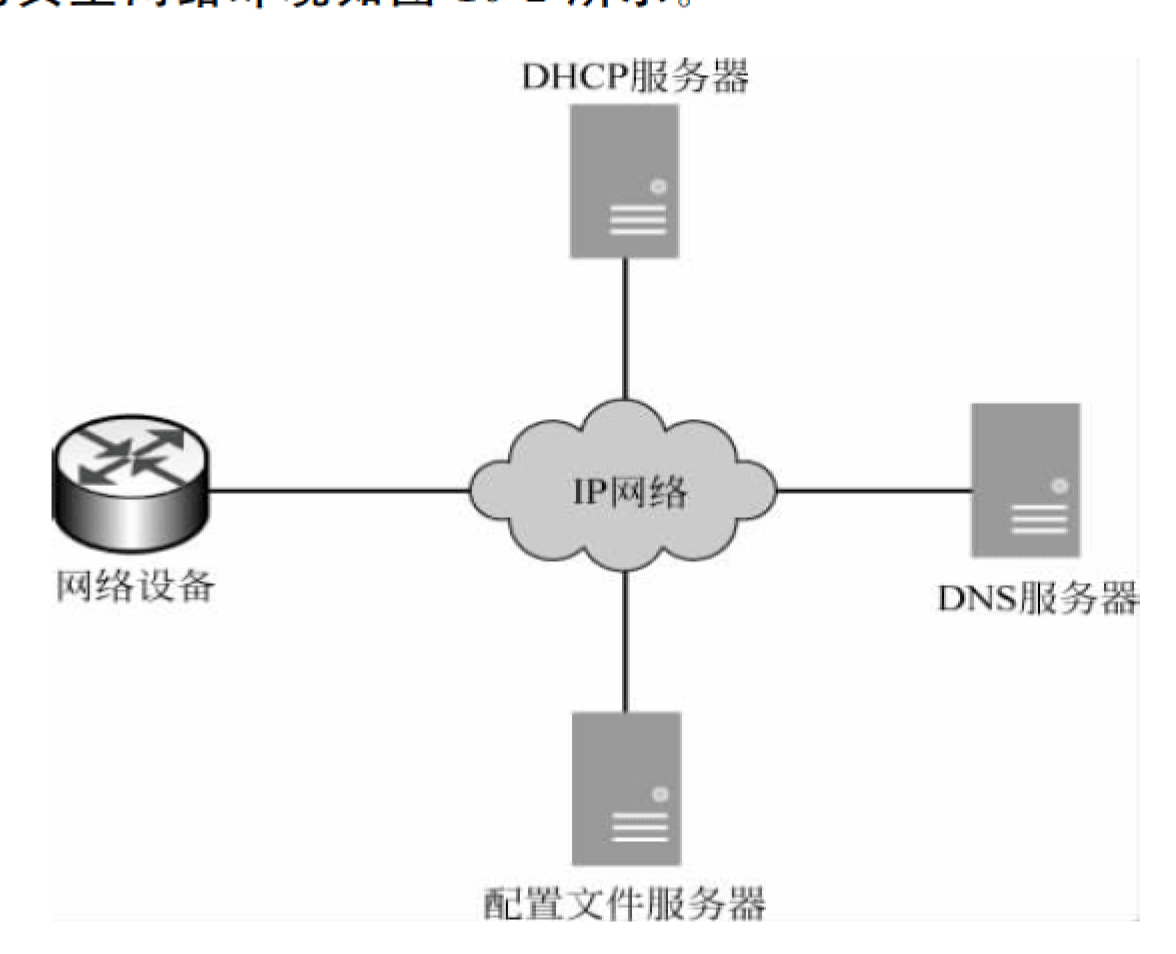


图 10-2　自动配置的典型网络环境

自动化配置的步骤如下：

（1）配置文件服务器。设备可以通过 HTTP 服务器或 TFTP 服务器获得配置文件，用户需要根据选用的方式在文件服务器上配置相应的 HTTP 服务或 TFTP 服务。设备

从文件服务器上获取的文件类型分为配置文件与配置脚本两种。

(2) 配置 DHCP 服务器。DHCP 服务器为执行自动配置的设备分配 IP 地址，并对通告设备获取配置文件或配置脚本的途径。DHCP 服务器可以根据用户需要的配置文件类型进行相应的配置(配置脚本和配置文件实现一致)。

(3) 配置 DNS 服务器。在使用服务器自动配置功能时，有两种情况用户需要配置 DNS 服务器：当 TFTP 服务器上不存在对应的主机名文件时，执行服务器自动配置的设备可以通过 DNS 服务器将自己的 IP 地址解析为主机名，以便从 TFTP 服务器获取到配置文件；如果设备从 DHCP 应答报文中获取到 TFTP 服务器的域名，则设备还可以通过 DNS 服务器将 TFTP 服务器的域名解析为 TFTP 服务器的 IP 地址。

(4) 配置网关。如果 DHCP 服务器、文件存储服务器、DNS 服务器、执行服务器与网络设备不在同一网段，这时就需要部署网关设备，使得每个服务器和设备之间路由可达，并在网关上配置 DHCP 中继功能。

(5) 选择接口获取配置文件。用户将设备的管理以太网接口连入网络中，这样可以加快服务器自动配置的速度。如果设备当前不存在配置文件，设备即可自动执行服务器自动配置流程。

(6) 完成配置过程。网络设备获取并执行配置文件成功，则整个自动配置过程结束；如果执行配置文件失败，则设备会在一定时间后开始执行下一次服务器自动配置流程，用户可以手动终止自动配置操作。设备通过服务器自动配置获取到的配置文件执行完成后，该文件将被删除，不会保存。

10.1.3 基于 Web 的网络管理技术

上述两种技术极大地方便了对网络的管制管理，但还是对网络管理人员的技术要求较高，不能直观方便地对网络进行管理。传统的网络管理平台是专用的网络管理平台，价格昂贵，对网络管理人员的素质要求较高，而且系统复杂。随着 Web 技术的成熟，网络管理系统的实现有了新的选择，就是不要求专用客户端、界面友好且可以动态扩展和更新的基于 Web 的网络管理系统。将 Web 技术引入网络管理中，可以使用户方便简单地管理网络和系统，并且很容易更新和定制，满足不同的需求。基于 Web 的网络管理主要提供两方面的功能：一是利用 Web 服务器、浏览器和其他 Internet 技术，对网络和系统进行管理、监视、维护和报告；二是利用 Web 浏览器技术改进系统管理功能。

基于 Web 的网络管理技术与其他的网络管理技术相比，在管理与用户界面方面有着独特的优势。

(1) Web 提供了更加友好的界面，统一的浏览器界面方便了用户的使用和学习，从而节省了管理开销和培训费用。

(2) 无须安装和配置任何管理软件，可以在 Internet 的任何站点上利用 Web 浏览器访问网络管理信息。

(3) 基于 Web 的系统提供的信息与传统的方法相比具有更高的层次，因此用户可以无须了解诸如管理信息的结构等细节，而能够直接获得综合的、反映本质的管理服务。

(4) 由于管理应用程序独立于平台，可以通过标准的 HTTP 或者 HTTPS 协议将多

个基于 Web 的管理应用程序集成在一起，实现管理应用程序间的方便访问。

(5) 基于 Web 的网络管理融合了 Web 的优势与传统的网络管理技术，是一种更加强大与方便的管理方式。

基于 Web 的网络管理基本模型[4]可以分为三部分：代理层、管理服务器层和客户端部分，如图 10-3 所示。

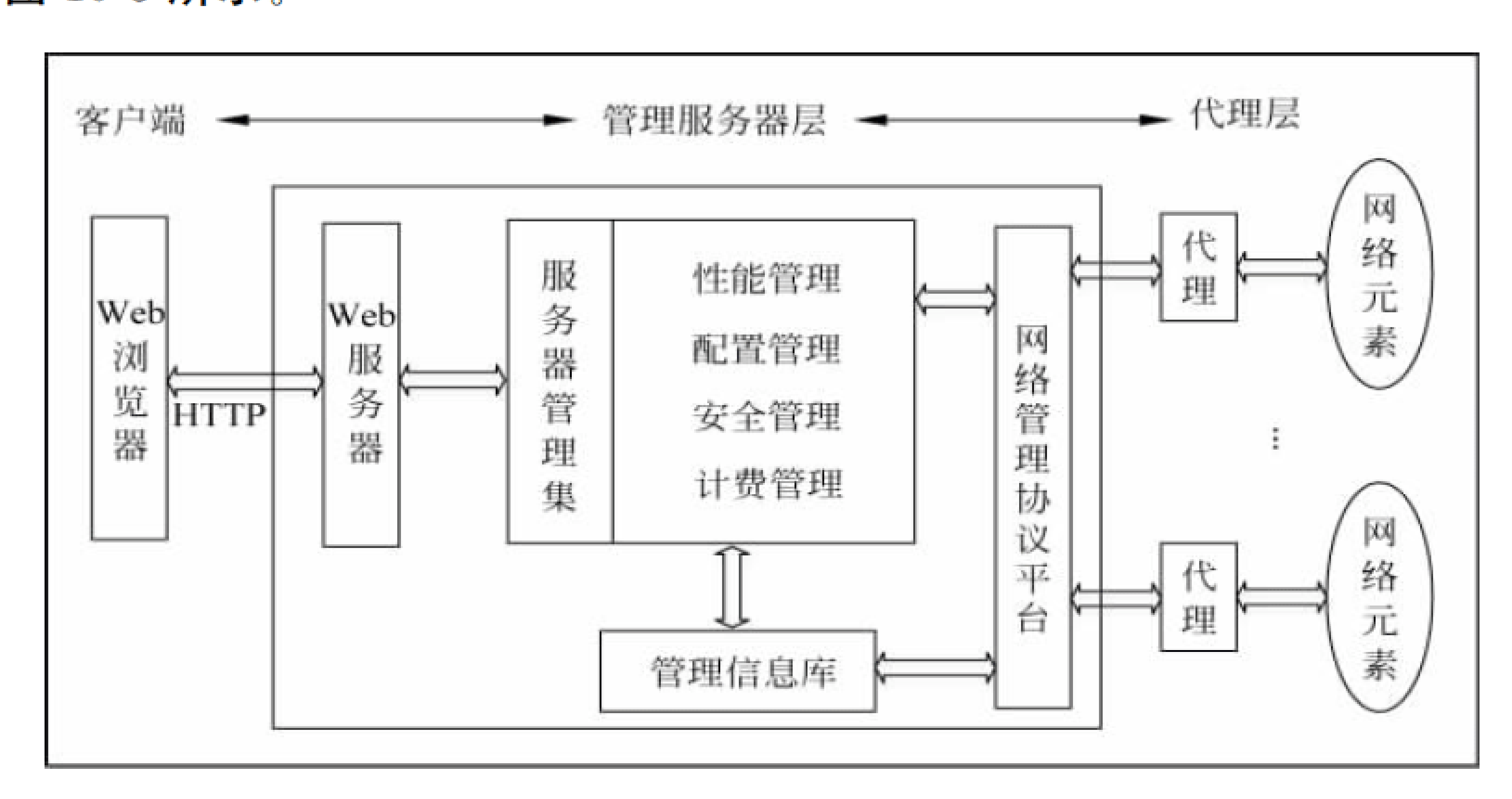


图 10-3 基于 Web 的网络管理模型

(1) 客户端主要提供一个基本的 Web 人机界面，用于完成具体网络管理的配置操作。Web 浏览器界面为网络管理员提供了一个统一、灵活的网络管理界面。

(2) 管理服务器层分为 Web 服务器与网络管理服务器两部分。Web 服务器作为浏览器与网络管理服务器的中间对接者。网络管理服务器为网络管理提供各种服务，如拓扑发现、网络配置、系统性能监控、故障定位与排查、安全保障与计费等。

(3) 代理层主要完成被管网络资源与业务的代理功能。例如，目前一些网络设备具有支持 SNMP 的内嵌代理系统，但是还有一些设备并不支持某些代理系统，因此使用不同的技术开发相应的代理服务程序（中间件）是实现网络管理的重要环节。

10.3 节所介绍的绿色中继异构网络实验平台即是基于 Web 的网络管理系统。该系统采用 Java 作为开发语言，开发管理服务器层；使用前端技术 HTML、JavaScript、CSS 以及 JSP 开发用户浏览器界面；对于代理层中间件的开发，使用 OpenWrt 官方所建议使用的脚本语言 Lua，以此可以充分利用 OpenWrt 官方提供的 API，减少工作量。详细开发过程见第 10.3 节。

10.2 OpenWrt 编译安装与开发

10.2.1 OpenWrt 简介

OpenWrt 是一个用于嵌入式设备的 GNU/Linux 发行版，具有强大的扩展性与网络组件。OpenWrt 支持各种处理器架构，无论对 ARM、×86、PowerPC 还是 MIPS 都有很好的支持[5]。不同于其他许多用于路由器的发行版，OpenWrt 是一个从零开始编写的、

功能齐全的、容易修改的路由器操作系统。实际上，这意味着用户能够使用自己想要的功能而不加进其他的累赘，而支持这些功能工作的 Linux kernel 又远比绝大多数发行版来得新。OpenWrt 附带有 3000 个左右的软件包，用户可以方便地使用自定义功能来制作固件与移植各类功能到系统中。

OpenWrt 的软件包来源分为两种：官方下载、第三方软件包。

（1）官方下载。官方的 OpenWrt 软件包可以在软件仓服务器的 packages 文件夹下找到。在各个 packages 文件夹下可以找到的软件包只包含 shell 脚本，因此当然是体系无关的，而所有包含二进制文件的软件包都是体系相关的，并且其中一些，如 kmods，是需要特定内核版本支持的。每一个发行版本的目录下都包含针对不同的支持平台的子文件夹，每一个平台目录包含针对不同目标编译好的固件和一个包含可安装的 *.ipk 文件的 packages 目录。

（2）第三方软件包。第三方软件包没有经过测试且不被 OpenWrt 官方支持，对于它们的安全性和可用性没有任何担保。它们很有可能包含后门之类的东西。因此尽量从你信任的源安装软件包。

对于软件包的管理，OpenWrt 使用 opkg 命令，其中常用的命令如表 10-1 所示。

表 10-1　opkg 常用命令

opkg	打印所有有效命令和选项的列表
opkg update	下载现在有效的软件包的列表
opkg list	查看现在软件包列表中所有软件包的清单，可以使用正则表达式过滤
opkg list_installed	查看已安装软件包的清单
opkg install umurmur	安装名为 umurmur 的软件包

OpenWrt 系统的所有配置文件皆位于/etc/config/目录下。每个文件与它所配置的那部分系统功能相关。可用文本编辑器、UCI 命令行实用程序或 OpenWrt 所支持的各种语言（Shell、Lua、C）API 来编辑、修改这些配置文件。OpenWrt 路由系统基本配置文件如表 10-2 所示，与其他业务相关的配置文件可以自行去官网查看。

表 10-2　OpenWrt 系统基本配置文件

配置文件	功能
Dhcp	dnsmasq 和 DHCP 的配置
Dropbear	SSH 服务端选项
Firewall	中央防火墙配置
Network	交换、接口和路由配置
System	杂项与系统配置
Timeserver	rdate 的时间服务器列表
Wireless	无线设置和无线网络的定义

接下来以/etc/config/wireless 无线设置配置文件为例来简单说明配置文件的基本语法。

```
config wifi - device 'radio0'
option type 'mac80211'
option hwmode '11g'
option path 'pci0000:00/0000:00:01.0'
option htmode 'HT20'
option txpower '20'
option country '00'
option channel '1'
config wifi - iface
option device 'radio0'
option ssid 'OpenWrt'
option encryption 'none'
option network 'lan'
option mode 'ap'
option wds '1'
```

其中，语句 config wifi-device 'radio0'语句标志着一个节的开始。这里的配置类型是 wifi-device，配置名是 radio0。配置中也允许出现匿名节，即自定义了配置类型而没有配置名的节，如文件第二节。配置类型对配置处理程序来说是十分重要的，因为配置程序需要根据这些信息来处理这些配置项，所以配置类型不能匿名。语句 option device 'radio0'、option ssid 'OpenWrt'等是配置文件中选项定义方式。其中，option device 'radio0'表示定义 wifi-iface 的名字为 radio0，option ssid 'OpenWrt'表示 wifi 的 ssid 为 'OpenWrt'。可以通过登录路由系统自带的 Web 页面，来查看对路由系统可视化操作中所对应的配置文件中的更改。

10.2.2 OpenWrt 编译与安装

OpenWrt 编译环境是 Linux，这里采用在虚拟机上安装 Ubuntu 的方式来搭建编译环境。在安装 Linux 系统时，需要注意相应系统位数的问题。

接下来需要搭建编译环境，具体命令如下。

```
sudo apt - get install gcc ;
sudo apt - get install g++;
sudo apt - get install binutils ;
sudo apt - get install patch ;
sudo apt - get install bzip2 ;
sudo apt - get install flex ;
sudo apt - get install bison ;
sudo apt - get install make ;
sudo apt - get install autoconf ;
sudo apt - get install gettext ;
sudo apt - get install texinfo ;
sudo apt - get install unzip ;
sudo apt - get install sharutils ;
sudo apt - get install subversion ;
sudo apt - get install libncurses5 - dev ;
sudo apt - get install ncurses - term ;
```

```
sudo apt-get install zlib1g-dev ;
sudo apt-get install gawk ;
sudo apt-get install asciidoc ;
sudo apt-get install libz-dev ;
```

编译环境搭建成功以后，使用命令 mkdir OpenWrt 创建一个文件夹，用来存放编译 OpenWrt 的源码。使用命令 cd OpenWrt 进入 OpenWrt 后，执行源码下载命令 svn checkout svn://svn.openwrt.org/openwrt/ * vision。 * vision 是想要下载的 OpenWrt 的对应版本。然后将下载后源码文件夹的权限更改为 777，所用命令为 chmod -R 777 OpenWrt。

编译之前还需要对软件包进行更新，命令为./scripts/feeds update -a。将更新的软件包显示在固件定制的界面上，命令为./scripts/feeds install -a。此时编译环境已经准备好，可以编译自己想要的固件了。

使用 make menuconfig 命令进入内核定制界面，如图 10-4 所示。

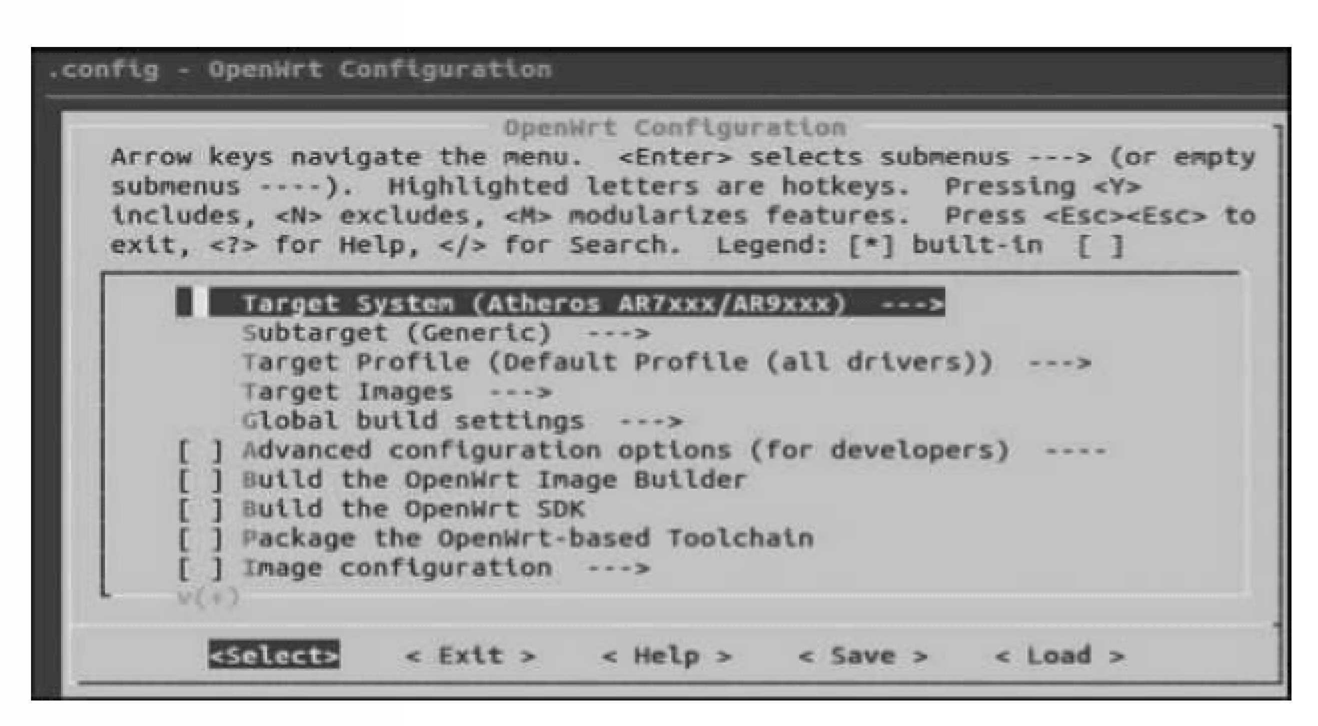


图 10-4　OpenWrt 固件定制主界面

在图 10-4 所示界面中可以选择型号、系统功能组件。需要用到方向键、回车键、空格键与 Esc 键。方向键是移动光标，回车键是确认，空格键是选择(在选项框按下空格键，* 号是编译进固件，M 是编译但是不编译进固件)，Esc 键用来返回。编译步骤如下：

(1) 选择 CPU 型号。在这个选项中，有很多型号的 CPU，每个型号的 CPU 在第二项 Target profile 有对应的芯片型号，可以根据情况进行选取。

(2) 选择路由型号。这项中会显示对应第一项 CPU 型号的路由种类。

LuCI 是 OpenWrt 自带的一种 MVC 的框架，包含很多可以方便地配置系统的接口，接下来的一节会详细地进行介绍。添加方式为选择 LuCI→Collections→luci 命令。

(3) 添加 LuCI 的中文语言包。为了方便进行研究学习，这里对 LuCI 安装中文语言包，方式为选择 LuCI→Translations→luci-i18n-chinese 命令。

可以根据自己的需要选择软件包加入固件，OpenWrt 的固件有 3000 多种，建议只选择自己所需要的进行添加，以免编译的固件过大，超出路由器规定的大小导致编译失败。若以后再需要某些功能的软件包，则可以通过 opkg 命令进行添加。如果仅仅需要一个可以使用的固件，则可以只选 CPU 型号与路由器型号，其余默认即可。

OpenWrt 固件编译好以后，将固件烧入路由器中，编译好的固件放在 bin 文件夹下。

可以从路由器 Web 界面中的对应选项烧入 OpenWrt，不同的原始路由系统入口界面可能不同，也可以重新刷入固件，对现有系统进行更新。

刷入以后，可以登录 Web 页面进行查看，看是否刷入成功。如图 10-5 所示，可以看到目前系统已经是所刷入的 OpenWrt 系统。

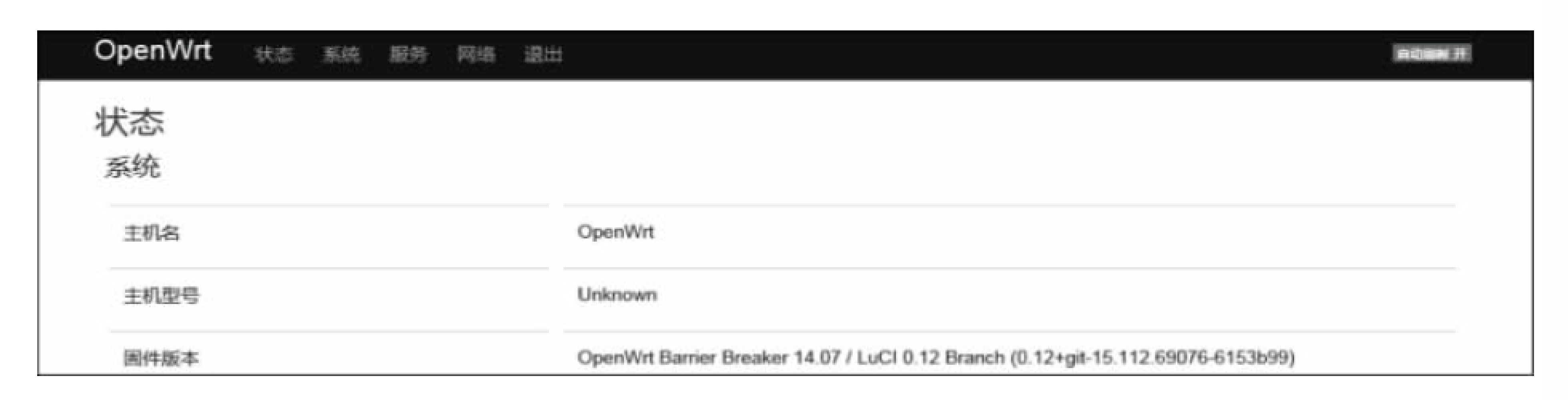


图 10-5　系统信息查看

10.2.3　LuCI API 的使用

前面介绍了 OpenWrt 的编译安装过程，这一节将简要介绍如何使用 OpenWrt。OpenWrt 官方提供了 OpenWrt 的开发工具 LuCI。LuCI 诞生于 2008 年 3 月，目的是为 OpenWrt 固件实现快速配置提供方便，LuCI 是 Lua 与 UCI 的合体。Lua 是一个小巧的脚本语言，很容易嵌入其他语言。轻量级 Lua 语言的官方版本只包括一个精简的核心和最基本的库。这使得 Lua 体积小、启动速度快，从而适合嵌入在别的程序里。UCI 是 OpenWrt 中为实现所有系统配置的一个统一接口，英文全称为 Unified Configuration Interface，即统一配置接口，LuCI 可以实现路由的网页配置界面。

LuCI 框架是一个基于 MVC 模式的嵌入式设备 Web 框架，可以使用 WinSCP 软件查看 OpenWrt 的 Web 系统文件，如图 10-6 所示。

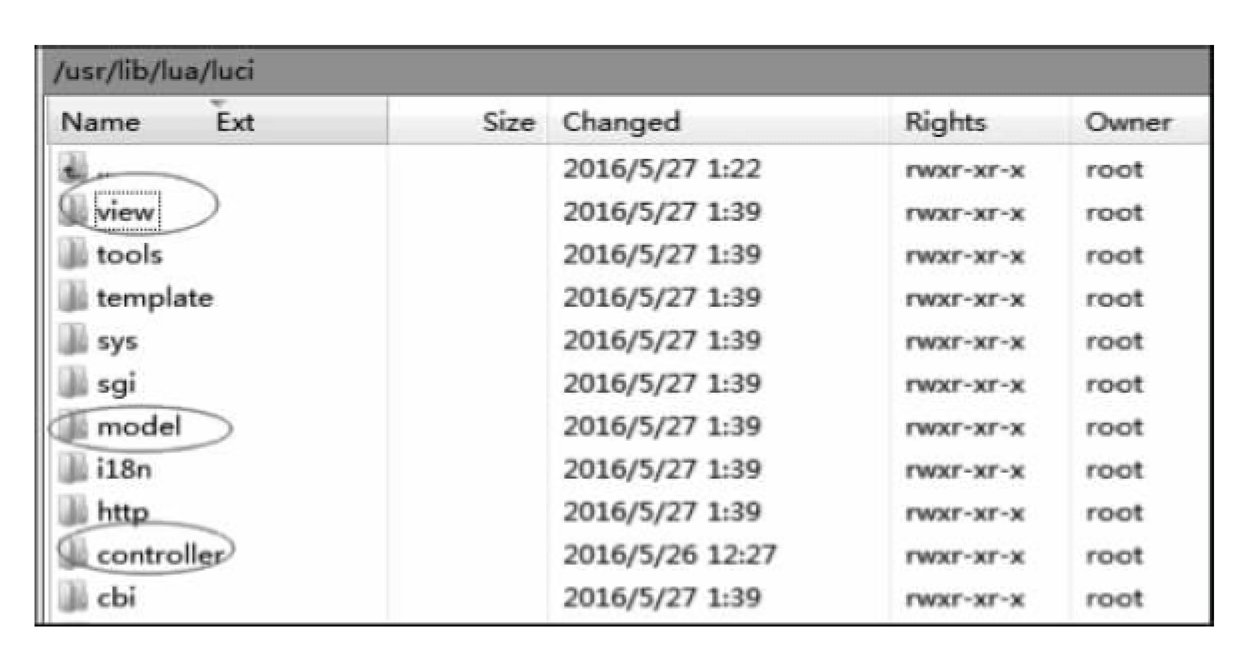


图 10-6　OpenWrt Web 系统

从图 10-6 中可以看到三个文件夹：view、model、controller，分别是 OpenWrt 系统的视图文件、模型文件与逻辑控制文件。OpenWrt 利用 uhttpd 作为 Web 服务器，实现客户端 Web 页面配置功能。对于 request 处理方式，采用的是 cgi，而所用的 cgi 程序使用 Lua 语言编写。

由于要建立一种集中式的设备管理平台，并不将用于控制的 Web Server 搭建在 OpenWrt 开发板上，所以对于 OpenWrt 自身的 Web 架构并不做过多深入研究，更加关注的点在于 LuCI 所提供的 API。使用 LuCI 所提供的 API 会大大地方便对系统的配置管理。

通过 LuCI 官方 API 网站，可以去学习了解所提供的 API 的作用与用法。如项目中需要经常用到的对系统进行配置、获取系统参数的功能，相应函数封装于 luci. model. uci 模块中。用于获取系统参数的 API 函数为 Cursor：get（config，section，option），config 参数为配置模块名称，section 为配置模块中对应参数所在节名，option 则是对应的参数名称。对于 OpenWrt 系统配置来说，需要组合使用三个 API 函数，分别是 Cursor：set（config，section，option，value）、Cursor：save（config）、Cursor：commit（config），后面会举例说明用法。

10.2.4 OpenFlow 添加与应用

OpenWrt 系统不但支持开发者针对设备本身管理方面的开发实践，而且随着 SDN 越来越成熟，OpenWrt 推出了 OpenvSwitch 组件，用来支持 OpenFlow 协议。这里介绍一下如何在 OpenWrt 系统中进行 OpenvSwitch 的安装，以及如何进行 OpenFlow 协议的验证，主要包括所用控制器 pox 介绍、如何安装 OpenvSwitch，以及 OpenvSwitch 与控制器的连接。

1. pox 控制器

pox 是一种基于 OpenFlow 的控制器，能将交换机送上来的协议包交给指定软件模块。pox 主要由 core 和组件组成，其中 core 中比较重要的模块是 of_01 和 openflow。of_01 主要是运行一个线程，该线程不断与交换机进行 TCP 连接，当某交换机送来一个协议消息时，of_01 会触发该消息所对应的事件。openflow 则与所有的物理交换机相连，而控制器可以通过 openflow 控制所有的交换机。在 Ubuntu14. 04 环境下，使用命令 sudo git clone http://github. com/noxrepo/pox 直接获取并安装 pox。

2. OpenvSwitch 安装部署

OpenvSwitch 是一种运行在虚拟化平台（如 KVM、Xen）上的虚拟交换机。OpenvSwitch 提供了对 OpenFlow 协议的支持，用户可以使用任何支持 OpenFlow 协议的控制器对 OpenvSwitch 进行远程管理控制。在 OpenvSwitch 中，有如下几个与 OpenFlow 密切相关的特性：

（1）Bridge——Bridge 代表一个以太网交换机（switch），一个主机中可以创建一个或者多个 Bridge 设备。

（2）Port——端口，与物理交换机的端口概念类似，每个 Port 都隶属于一个 Bridge。

（3）Interface——连接到 Port 的网络接口设备。在通常情况下，Port 和 Interface 是一对一的关系，在配置 Port 为 bond 模式后，Port 和 Interface 可以是一对多的关系。

（4）Controller——OpenFlow 控制器，OpenvSwitch 可以同时接受一个或者多个

OpenFlow 控制器的管理。

向 OpenWrt 中添加 OpenvSwitch 如添加其他安装软件包一样，可以在 OpenWrt 系统编译时添加 OpenvSwitch 软件包，或者使用 opkg 命令进行单独安装。但是安装过程中有些问题需要注意。添加 OpenvSwitch 功能时，attitude adjustment 和 trunk 两个版本 OpenWrt 系统的处理方式不同，对于 trunk 版本的 OpenWrt，官方给定软件套件里已经包含了 OpenvSwitch 这个软件，只需要执行 feeds 更新，再选择 OpenvSwitch 后执行编译即可。下面详细说明 attitude adjustment 版本的编译过程。

在 attitude adjustment 版本的官方软件包套件（也就是 feeds. conf. default 里列出的那些软件源）中是没有包含 OpenvSwitch 的，需要用户自己添加相应的软件源。使用命令：

```
cd $ TOPDIR
echo "src - git openvswitch git://github.com/schuza/openvswitch.git" > feeds.conf.default
```

即可完成添加，接下来进行 OpenWrt 系统的正常编译即可。

3. 连接 pox 控制器与 OpenvSwitch

1）启动 pox 并开始监听

一台计算机以无线方式连接 OpenWrt 路由器，假设路由 IP 地址为 192.168.1.1，该计算机 IP 地址为 192.168.1.143，操作系统为 Ubuntu 14.04，并已安装 pox。在 pox 目录下使用命令：

```
./pox.py openflow.of_01 --address=192.168.1.143 --port=9999 forwarding.l2_learning
```

以启动 pox 并开始监听。其中，pox. py 是程序的入口；需要 openflow. of_01 库解释后面的参数；--address 后面参数为 pox 所在主机的 IP 地址；--port 后面的参数指定 pox 所要监听的端口，不指定则默认为 6633，也可以自己指定，这里指定 pox 监听 9999 端口；forwarding. l2_learning 为 pox 提供的组件。开启成功如图 10-7 所示。

```
z@ubuntu:~/pox$ ./pox.py openflow.of_01 --address=192.168.1.143 --port=9999 forwarding.l2_learning
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:core:POX 0.2.0 (carp) is up.
```

图 10-7　pox 开启界面

2）在 OpenWrt 中发起连接请求

使用另一台计算机以无线方式连接 OpenWrt 路由器，然后通过 ssh 方式登录 OpenWrt 系统。登录成功后使用命令 ovs-vsctl add-br br0 新建一个 bridge：br0，使用命令 ovs-vsctl add-port br0 eth0 向 br0 添加一个 port：eth0，使用命令 ovs-vsctl set-controller ovs-br tcp:192.168.1.143:9999 向控制器发起连接请求，如图 10-8 所示。

```
root@OpenWrt:~# ovs-vsctl add-br br0
root@OpenWrt:~# ovs-vsctl add-port br0 eth0
root@OpenWrt:~# ovs-vsctl set-controller br0 tcp:192.168.1.143:9999
root@OpenWrt:~#
```

图 10-8　OpenvSwitch 发起连接请求

当 OpenWrt 系统路由器发起请求后，位于 IP 地址为 192.168.1.143 计算机上的 pox 立即监听到请求并连接成功，如图 10-9 所示。

```
z@ubuntu:~/pox$ ./pox.py openflow.of_01 --address=192.168.1.143 --port=9999 forwarding.l2_learning
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:core:POX 0.2.0 (carp) is up.
INFO:openflow.of_01:[00-02-2b-00-1b-c7 1] connected
```

图 10-9　pox 连接成功界面

此时，在 OpenWrt 中可使用命令 ovs-vsctl show 查看当前信息，如图 10-10 所示，图中信息表示 bridge br0 已成功连接到指定的控制器。

```
root@OpenWrt:~# ovs-vsctl show
0106f445-570d-475f-9625-293758d1ee92
    Bridge "br0"
        Controller "tcp:192.168.1.143:9999"
            is_connected: true
        Port "eth0"
            Interface "eth0"
        Port "br0"
            Interface "br0"
                type: internal
root@OpenWrt:~#
```

图 10-10　当前 OpenvSwitch 信息

10.3　实验平台搭建

上面介绍了网络管理相关技术以及 OpenWrt 系统及其开发，这一节详细介绍如何搭建绿色中继异构网络实验平台。平台通过支持 OpenWrt 系统的路由模拟现网中的基站与中继，通过连接到网络设备上的终端（包括手机、笔记本、平板等终端）模拟基站与中继下的用户，通过太阳能电板模拟绿色能源。整个平台系统是一个基于 Web 的网络管理系统，在该实验平台中，管理人员可以通过运行在客户端上的管理界面，监测到每个网络节点的当前能源供给状态与本身状态，如上/下行流量、接入用户数量、发射功率大小。在客户端为网络节点配置好管理策略后，各网络节点能够根据当前能量存储状态、节点本身状态动态改变其发射功率、能源供给选择、用户分离和休眠，实现网络拓扑结构动态变化。

10.3.1　平台设计

整个平台按照功能可以设计为三个模块：集中式管理模块、组网模块与能源供给模块，如图 10-11 所示。集中式管理模块中包括网络的状态监测与状态控制两部分，它们全部服务于管理员所设定的策略。组网模块中不同节点采用不同的能源供给方式，可以通过功率检测仪监测能源供给与节点的能耗。

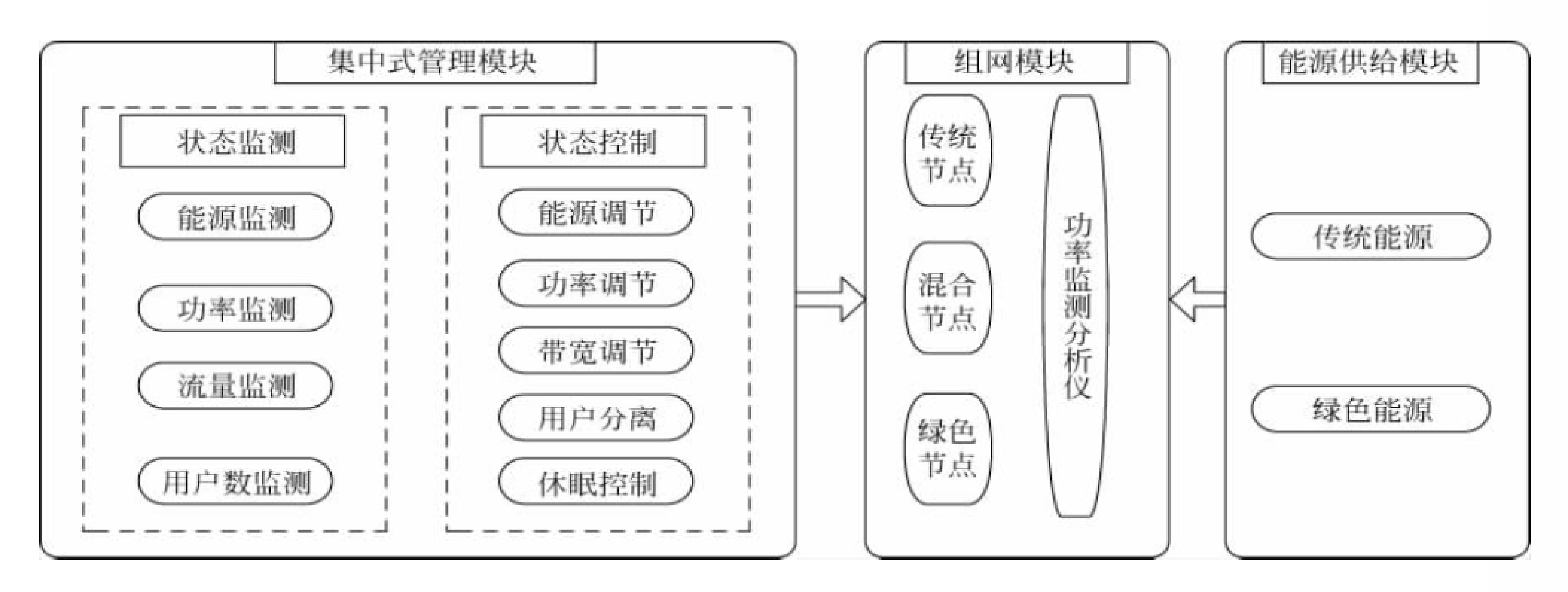


图 10-11　绿色中继增强异构网络实验平台

10.3.2　能源供给模块

绿色中继异构网络实验平台的能源供给模块中，有传统能源和绿色能源两部分。其中，绿色能源的能量来自太阳能电板，而传统能源的能量来自电网。能源供给模块的主要功能除了向网络节点供能之外，还能将自身当前的能量存储状态实时上报给集中式管理模块，为网络设备的管理策略提供依据。

平台中采用型号为 20-18-P 的单晶硅太阳能电池板采集太阳能充当绿色能源，它的最大系统电压为 1000V，短路电流为 1.21A，开路电压为 21.67V，最大功率为 20W。采用型号为 OT 7-12 的蓄电池作为绿色能源存储设备，其标准输入为 12V/6A。由于太阳能电池板输出电压与电流不稳定，所以充电过程需要通过变压控制器作为中间转接来完成。同时，使用蓄电池向开发板供电时，由于电压与电流不相同，同样需要使用变压控制器的中间转接。

对于能源状态的监测，采用的是型号为 APN1211A-U 的功率分析仪。APN1211 系列单相功率分析仪是便携式轻巧设计，能测量交流、半波整流和直流负载，如 LCD 监视器等信息设备、绿色计算机、环保监视器、开关电源供应器、电动工具、信息及办公设备等相关产品，也适用于对电网运行质量进行监测及分析，提供电力运行中的谐波分析及功率质量分析，可通过外接互感器实现更大的负载测量，能够对电网运行进行长期的数据监测。同时配备 PC 端数据分析软件，对上传至计算机的测量数据进行各种分析、保存、打印等。所以可以使用功率分析仪获取路由器的功率，并且能够捕获不同负载下其功率变化情况。

如图 10-12 所示为实验所搭建的能源状态监测装置。使用太阳能电池板收集的能源通过变压控制器存储到蓄电池中，然后通过变压控制器将电能输出给路由器，同时使用功率分析仪对输出功率进行采集保存，以供集中式管理模块下发控制指令时调用。



图 10-12　绿色中继增强异构网络平台能耗监测

10.3.3　组网模块

组网模块中，可以使用支持 OpenWrt 路由系统的路由器或开发板构建物理网络。路由器可以选择 HuaweiHG556a，其具有优良的性能，包括 300Mb/s 高速 CPU，64MB 内存，300MB 无线，支持 USB 挂载与 3G 网卡。开发板可以选择 MT7620 系列，MT7620A 与 MT7610E Wi-Fi 路由器同时支持最新的 1T1R 802.11ac (433Mb/s) 及 2T2R 802.11n (300Mb/s) 标准，可让用户轻松建立总传输速度高达 733Mb/s 的同步双频无线网络环境，使信息、语音及高画质影音传输各走各的频段而不互相干扰，带来最佳的用户体验。MT7610U USB 及 MT7610E PCIe 单芯片解决方案则适用于移动及嵌入式装置，并且 MT7620 系列路由器能较好地支持 OpenWrt 系统，满足开发要求。

组网模块网络设备采用不同的能源供电，其中包括使用传统能源供电、传统能源与绿色能源混合供电以及绿色能源供电。网络中每个设备职能不一，模拟真实网络中不同类型的网络设备组成，各个网络设备之间相互独立，每一个设备的发射功率、带宽、工作模式、能源供给均可以通过管理平台来监测和控制。模拟基站节点与中继节点的路由器处于级联模式，所有网络设备构成了一个局域网。

对于模拟中继蜂窝网络的搭建，可以通过有线与无线两种方式进行。有线方式搭建的中继网络中模拟基站的网络设备一侧通过有线与 Internet 相连，另外一侧通过有线方式与模拟中继的网络设备相连，模拟基站的设备与模拟中继的设备同时为其所覆盖范围内的终端提供服务。有线方式的搭建较为简单，这里不做叙述。下面详细介绍一下无线方式的构建方法。

这里使用两个 OpenWrt 系统的路由器进行演示，选择其中一个路由器作为主路由器模拟基站，另一个作为中继路由器模拟中继站。其中，主路由器无线配置情况如图 10-13 所示，它的 SSID 为 OpenWrt1，接着将它的 LAN 口 IP 设置为 192.168.1.1。

接下来，将另一个路由器设置为主路由器的中继路由。首先需要将中继路由的 IP 网络设置为与主路由器不同，如图 10-14 所示。



图 10-13　主路由器无线概况　　　　图 10-14　中继路由 IP 设置

接下来在中继路由无线概况处，单击“搜索”按钮，如图 10-15 所示。

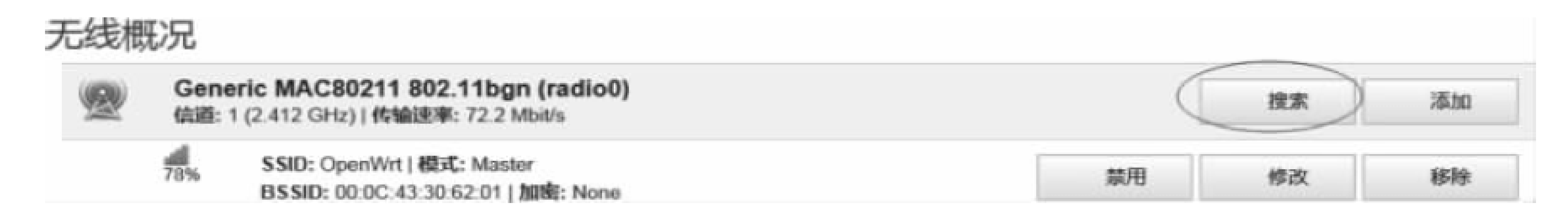


图 10-15　中继路由无线概况

在接下来的界面中，会显示中继路由可以搜索到的无线网络，选中所要加入的 SSID 为 OpenWrt1 的无线网络，如图 10-16 所示。



图 10-16　中继路由网络搜索

如果上级网络的接入需要密码，接下来会出现密码输入窗口。若没有密码，则直接进入到网络设置界面，如图 10-17 所示。



图 10-17　中继路由网络设置

接下来会跳转到无线配置界面，如图 10-18 所示。可以看到，中继路由无线配置情况与主路由相同，单击确认提交。



图 10-18　中继路由无线配置

这时中继路由就已设置完成。使用有线方式接入 192. 168. 2. 1，在图 10-19 所示的接口页面可以看到目前中继路由已经从主路由器获取了 IP。



图 10-19　中继路由设置完成

接下来再创建一个无线接口作为无线接入点，连接终端。单击无线页面中的“添加”按钮，填写设置参数，创建新无线接口，如图 10-20 所示。



图 10-20　创建新无线接口

从图 10-20 中可以看到，无线概况处已经有两个无线接口信息，其中一个是处于中继路由模式的 OpenWrt1，还有一个是处于接入点模式的 OpenWrt。

10. 3. 4　集中式管理模块

实验平台中的集中式管理模块主要实现对能源供给模块和组网模块的集中式管理。通过对能源状态和各个节点运行状态的监测，在相关的网络管理机制的约束下，对各个网络节点的能源供给、发射功率、休眠与否进行实时配置。

对于网络设备本身运行状态的监测，首先利用 LuCI 所提供的 API 函数或者通过 Shell 编写脚本获取相关参数。然后在服务器上搭建数据库，可以使用 MySQL 或者其他数据库。将脚本按时循环执行获取网络设备运行数据，存入到已经建立数据库的对应表中。对于不同的策略的制定，可能需要不同的参数。如节点休眠策略，可能需要节点用

户数、流量负载等参数。对于网络状态控制，首先在前端界面传入配置参数，然后在 Web 服务器接收以后，由数据库中获取数据，经过运算后将要配置的参数传给设备，设备上的服务器程序接收参数以后，执行动作完成配置。集中式管理模块由以下三部分组成：设备端、Web 服务器与客户端。

1. 设备端

设备端程序按照功能划分主要分为以下三部分：数据采集存储程序、与服务器端通信程序和被调用用于执行操作的程序。其中，数据采集程序主要采用 LuCI 提供的 API 与 Shell 编写脚本程序，将定时获取的数据存入到 MySQL 数据库创建的表中。

MySQL 是一个开放源码的小型关联式数据库管理系统[4]，由于其体积小、速度快、总体拥有成本低并且开放源码，使其应用广泛。MySQL 安装方便，可以在 Windows 或者 Linux 上快速搭建 MySQL 数据库，并通过安装 MySQL 客户端对数据库进行直观的观察与管理。下面以某个节点的功率采集存储为例做演示。

首先登录 MySQL，在 MySQL 中建立名为 ap_manage 的数据库，命令为 create database ap_manage，然后执行 use ap_manage，将 ap_manage 设置为当前所用数据库。接下来将在 ap_manage 数据库中创建 power 表用来存储所采集的节点功率，命令如下：

```
create table power(
time varchar(20),            //存入时间
valuevarchar(20)            //存入数值
)
```

创建完成后可以使用命令 desc power 来查看所创建的表是否是按照期望的方式创建的，如图 10-21 所示。

Field	Type	Null	Key	Default	Extra
time	varchar(20)	YES		NULL	
value	varchar(20)	YES		NULL	

图 10-21 数据表信息

也可以安装 MySQL 的可视化客户端，对表进行管理与查看，如 Navicat。Navicat 是一套快速、可靠的数据库管理工具，它提供多达 7 种语言供客户选择，被公认为全球最受欢迎的数据库前端用户界面工具，可以用来对本机或远程的 MySQL、SQL Server、SQLite、Oracle 及 PostgreSQL 数据库进行管理及开发。如图 10-22 所示是通过 Navicat 对数据库进行查看的示例。



图 10-22 Navicat 信息查看

也可以通过 Navicat 直接操作数据库中表，对其进行修改等操作。如果在创建时没有对表的主键进行设置，则通过 Navicat 进行设置。单击图 10-22 中“设计表”按钮，会出现图 10-23 所示内容，单击 time 栏最后方格，出现黄色主键标志，就可以将 time 栏设置为主键。也可以通过其他选项来完成其他操作，如更改数据类型等。

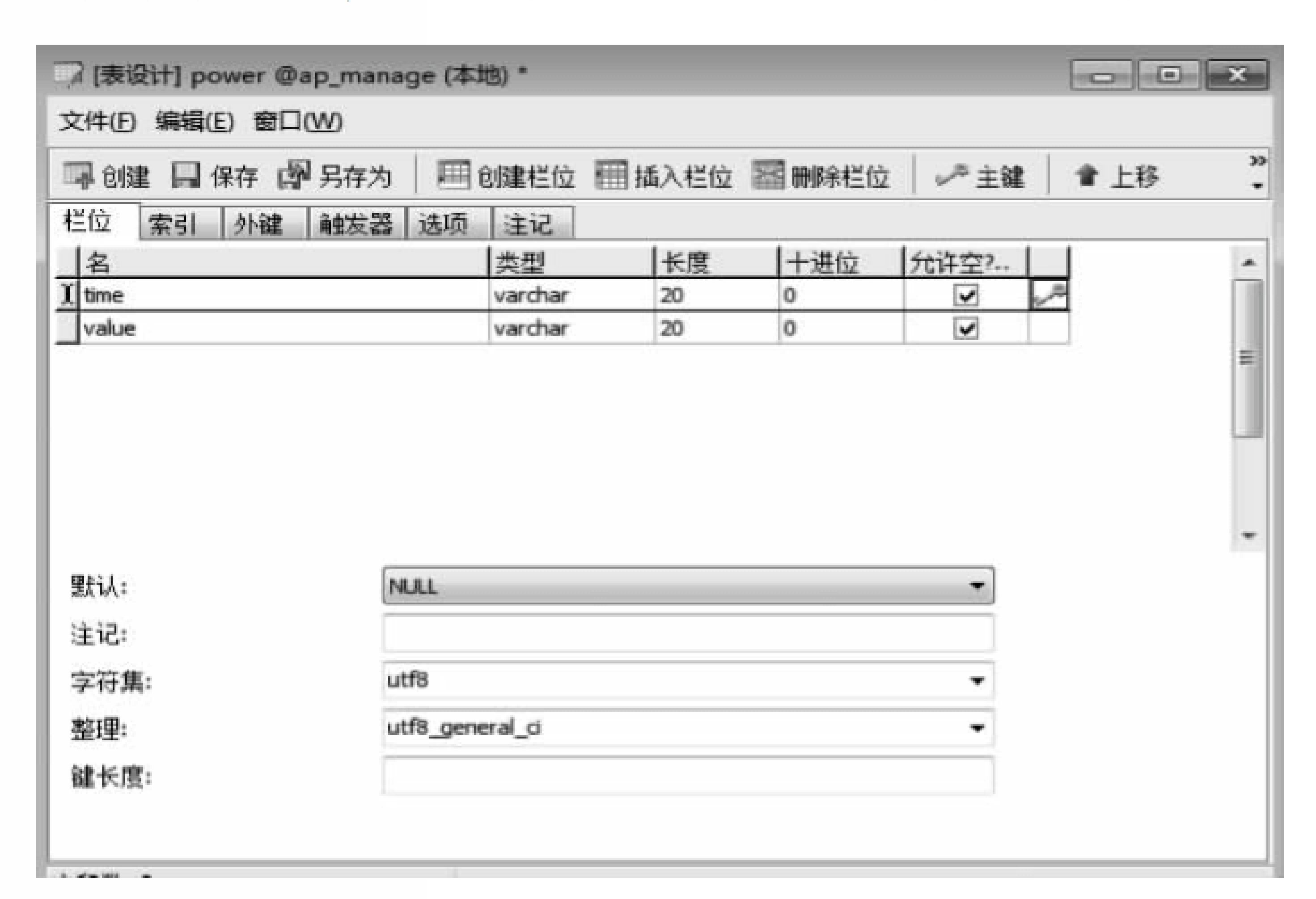


图 10-23　Navicat 设置表属性

这时再通过命令行查看表，可以看到表的属性已经更改，如图 10-24 所示。

```
mysql> describe power;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| time  | varchar(20) | NO   | PRI |         |       |
| value | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

图 10-24　更改后表信息

创建数据库与表以后，接下来就可以在设备端编写数据采集存储程序，用于采集设备的功率参数并进行存储。其中，数据采集采用 LuCI 提供的 API 函数，代码如下：

```
local GetPower = require("luci.model.uci").cursor()
power = GetPower.get("wireless","radio0","txpower")
```

首先需要通过 require 函数引入 luci. model. uci 模块，然后实例化一个变量 GetPower，通过调用 get 函数，来获取无线接口 radio0 的发射功率。Get 函数采用的方式是获取配置文件中的值，其中 wireless 为配置类型，radio0 为配置名，txpower 为配置的节，get 所获取的为配置的值。

参数获取之后，需要将参数存入数据库中。首先需要导入 Lua 关于 MySQL 的软件包，语句为 mysql = require "luasql. mysql"。然后创建一个 MySQL 环境并建立该环境下的连接，语句如下：

```
local env  = mysql.mysql()
```

```
local conn = env:connect(database name,user name,password,IP,port)
```

建立连接以后，就可以对数据库数据进行 insert、update、delete 等操作。将数据存入数据库的代码语句如下：

```
conn:execute([[insertintopower values(time,value)]])
```

其中，time 为数据采集的时间，value 为所采集的数据值。随着数据采集时间延长或者采集密度的增加，表中的数据会越来越多，可以利用触发器，当数据数目到达一定程度，或者数据之间时间差过大时，将时间过于久远的数据删除。触发器是一种特殊的存储过程，它在插入、删除或修改特定表中的数据时触发执行，它比数据库本身标准的功能有更精细和更复杂的数据控制能力。

设备端与 Web 服务器采用 Socket 通信，在设备端启动一个 ServerSocket 程序用于监听，当 Web 服务器发起建立连接的请求时，实例化一个 Socket 对象，与 Web 服务器进行连接。代码如下：

```
socket = require("socket");
host = "前端服务器 IP";
port = "端口号";
server = assert(socket.bind(host,port));              //实例化 socket
while 1 do
local conn = assert(server:accept());
while 1 do
param,status = control:receive();                     //接收参数,监测 socket 状态
if f_type(param) == type1 then
        function_type1(f_value(param))
elseif f_type(param) == type2 then
        function_type2(f_value(param))
            …
end
end
end
```

程序中 param 为设备端接收到的参数，其中包含配置类型与数值。配置类型用来说明需要更改的配置，如功率、带宽等；数值用来说明需要更改为什么。可以编写函数 f_type(param)来提取接收参数中配置类型字段，f_value(param)用来提取配置值字段。function_type1、function_type2 分别对应不同配置类型下的不同操作函数。

对于操作函数的编写，同样可以使用 LuCI 所提供的 API。下面的代码是更改设备功率配置。

```
fuction (param)
local SetPower = require("luci.model.uci").cursor()
SetPower.set ("wireless","radio0","txpower",param)
SetPower.save ("wireless")
SetPower.commit ("wireless")
end
```

与参数的获取类似，同样是先实例化一个配置实例，然后调用 API 函数。一项配置

的更改,需要参数更改、保存、提交三步完成。

2. Web 服务器

Web 服务器程序运行以后,接收客户端传来的参数,向被控端计算机发起 Socket 连接指令。当与设备端连通之后服务器程序通过 Socket 向被控端计算机发出指令,设备端程序则根据指令完成一系列进程操作。

控制服务器程序采用 Java 编写,Java 是一种解释执行的程序语言,其解释器称为 JVM,即 Java 虚拟机。JVM 具有良好的可靠性,能运行于各种各样的环境。而且它还被捆绑到某些 Web 浏览器,从而使这些浏览器能够执行 Java 代码。Web 服务器程序主要分为以下三部分:执行特定操作的功能类、策略程序、中央控制器程序。其中用到的技术主要有数据库相关的 JDBC、作为中央控制器的 Servlet 与用来通信的 Socket 技术。

JDBC(Java data base connectivity,Java 数据库连接)是一种用于执行 SQL 语句的 Java API,可以为多种关系数据库提供统一访问,它由一组用 Java 语言编写的类和接口组成。有了 JDBC,向各种关系数据发送 SQL 语句就是一件很容易的事,程序员只需用 JDBC API 写一个程序就可以向不同的数据库发送 SQL 调用。同时,将 Java 语言和 JDBC 结合起来使程序员不必为不同的平台编写不同的应用程序,只需写一遍程序就可以让它在任何平台上运行。这也是 Java 语言“编写一次,处处运行”的优势,它还增进了访问数据的效率和快捷程度。JDBC 的使用可以分为三步:与数据库建立连接、发送操作语句、对获取的数据进行处理。与数据库连接的代码如下:

```
public GetConnection(String dbname,String username,String pwd){
    String driver = com.mysql.jdbc.Driver;
    String url = "jdbc:mysql://localhost:3306/" + dbname;
        try{
            Class.forName(driver);
            Connection conn = DriverManager.getConnection (url,user,pwd);
        }catch(Exception e){
            System.out.print("连接失败!!<br>" + e.toString());
        }
    }
```

函数 GetConnection 的功能是数据库连接,其参数中 dbname 是数据库名称,username 表示连接用户名,pwd 表示该用户登录数据库的密码。driver 是驱动程序名称,数据库连接首先需要加载驱动程序。

上面程序中 GetConnection 只是获取了与数据库的连接,从数据库中获得数据还需要执行相应的 SQL 语句。要执行 SQL 语言需要首先获得 Statement 类对象。通过上面代码中连接数据库对象 conn 的 createStatement()方法可以获得 Statement 对象。具体的代码示例如下:

```
publicint getmsg(String sql){
        try {
            Statementstmt = con.createStatement();
            ResultSetrs = stmt.executeQuery(sql);
        } catch (SQLException e) {
```

```
            e.printStackTrace();
            return 0;
        }
    }
```

有了 Statement 对象以后，可以调用相应的方法，执行 SQL 语句，对数据库进行查询与修改，并将查询的结果放入 ResultSet 的对象 rs 中。executeQuery 为 Java 程序中用来执行 SQL 语句的函数，String sql 是用来执行获取数据的 SQL 语句，sql = select * from power order by time，即从记录功率的表 power 中按照时间顺序获取不同时间的传输功率值。

Socket 的英文原义是“孔”或“插座”，通常也称作“套接字”，用于描述 IP 地址和端口，是一个通信链的句柄，可以用来实现不同虚拟机或不同计算机之间的通信。在 Internet 上的主机一般运行了多个服务软件，同时提供几种服务。每种服务都打开一个 Socket，并绑定到一个端口上，不同的端口对应于不同的服务。Java. net 包有 ServerSocket 用于服务器端，Socket 是在建立网络连接时使用的。ServerSocket 主要功能是用来等待网络上的请求，它可以通过指定的端口来等待连接的套接字，服务器端口一次可以与一个套接字连接。如果多台客户机同时请求连接，这时服务器套接字将会将请求放入队列，然后从中取出一个套接字与服务器新建的套接字连接起来。队列中存放的套接字数目大小可以设置，一般默认为 50。

当管理员有管理需求的时候，从 Web 服务器发送参数给设备，这时 Web 服务器需要创建一个 Socket 对象。Socket 类有构造函数 Socket(String host，int port)，构造函数创建一个基于 Socket 的连接设备端 Socket 的流套接字。Socket 创建代码需要用到 socket = new Socket(host，port)语句，host 是主机 IP，port 是 Socket 绑定的端口。Web 服务器创建了一个 Socket 对象后，可通过 getInputStream()方法从设备端服务程序获得输入流传送来的信息，也可通过调用 getOutputStream()方法获得输出流来发送消息。在读写活动完成之后，客户程序调用 close()方法关闭流和流套接字。可以将数据发送操作封装到一个类的方法中，在控制部分方便调用。代码如下所示，函数中参数 msg 是需要发送的信息。

```
public sendMsg(String msg){
try{
PrintWriter out = new PrintWriter(client.getOutputStream());
out.println(msg);
out.flush();
}catch(IOException e){
e.printStackTrace();
    }
}
```

这里采用 Servlet 程序作为中央控制器。Servlet(Server Applet)全称 Java Servlet，是用 Java 编写的服务器端程序[5]。其主要功能在于交互式地浏览和修改数据，生成动态 Web 内容。狭义的 Servlet 是指 Java 语言实现的一个接口，广义的 Servlet 是指任何实现了这个 Servlet 接口的类。一般情况下，将 Servlet 理解为后者。Servlet 运行于支持 Java

的应用服务器中。从实现上讲，Servlet 可以响应任何类型的请求，但绝大多数情况下 Servlet 只用来扩展基于 HTTP 的 Web 服务器。

Servlet 的工作模式为客户端发送请求至 Web 服务器(通常使用 tomcat 作为 Java 程序服务器)，服务器启动后调用 Servlet，Servlet 根据客户端的请求内容生成动态响应后再传递给服务器，服务器最后将响应内容返回给客户端。Servlet 功能与传统 CGI 类似，但是与传统 CGI 相比使用更加方便，功能也更加完善，并且可以移植到不同平台。

HTTPServlet 使用一个 HTML 表单来发送和接收数据。当表单提交信息时，它们指定服务器应执行哪一个 Servlet 程序。HttpServlet 类包含 init()、destroy()、service()、doGet()、doPost()等方法。其中，init()方法在 Servlet 的生命周期中仅执行一次，在服务装入 Servlet 时进行初始化；doGet()在当一个客户通过表单发出一个 HTTP GET 请求或直接请求一个 URL 时被调用，与 GET 请求相关的参数添加到 URL 的后面，并与这个请求一起发出。doPost()方法与 doGet()方法的不同之处在于当一个客户通过表单发出一个 HTTP POST 请求调用 doPost()方法时，与 POST 请求相关的参数作为一个单独的 HTTP 请求从浏览器发送到服务器。

例如，创建一个 Servlet 类，利用 doGet()函数从前端获取用户对某个设备的功率设定值，然后调用 Socket 连接与数据发送的方法传送到设备端。代码如下：

```
protected void doGet (HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
String parameter = null;
parameter = request.getParameter("power");
if(parameter == null || parameter.length() <= 0){
        //对于非法输入的处理，自行定义
        }else{
        //实例化一个 socket
        SocketClient client = new SocketClient("设备 IP",端口号);
        //向设备端发送参数
        client.sendMsg(parameter);
         //传送完毕后关闭 socket
    client.closeSocket();
        }
    }
```

对于网络管理策略的实施，可以另外建立 Servlet 用于接收客户端发送给调用某个策略的参数，来执行相应的策略。

3. 客户端

客户端用来提供给网络管理人员一个信息查看、配置下发的界面。网络管理人员可以直观地观察到网络中各个节点的配置情况，并且可以单独配置某个节点的某个参数，或者选择某条策略去管理整个网络。

对于客户端，采用技术主要是 HTML、JavaScript、JSP 等。HTML 指的是超文本置标语言，不是一种编程语言，而是一种标记语言，使用标记标签来描述网页。HTML 标记标签是由尖括号包围的关键词，通常成对出现，如<html></html>、<body></body>、

</br>等。HTML 文档就是网页，Web 浏览器可以读取 HTML 文档并以网页形式显示。当然，显示过程中浏览器不会显示 HTML 的标签，而是显示 HTML 标签内的内容。

其中最常用的标签有：<h1>～<h6>用来定义 HTML 文档标题；<p>用来定义段落；<a>用来定义连接，如一个连接可以表示为 This is a link ，单击网页上 This is a link 后，就会跳转到 w3school 的主页。HTML 标签与所包围的内容组成 HTML 元素，HTML 元素就是指从开始标签到结束标签的所有代码。上例中 This is a link 就是元素的内容。HTML 标签也可以拥有属性，属性提供了 HTML 元素的更多信息。属性以"名称='值'"的形式出现，如上例链接的地址在 href 属性中指定。

HTML 表单是一种被广泛应用的前端向后台提交数据的方式，使用表单标签<form></form>定义。表单是一个包含表单元素的区域，表单元素是允许用户在表单中输入信息的元素，如单选框、复选框、下拉列表、文本域等。

JavaScript 是属于 Web 浏览器端的一种脚本语言，用来验证表单、监测浏览器、创建 Cookies 等应用。JavaScript 可嵌入到 HTML 页面，目前几乎所有浏览器都能执行 JavaScript 代码。HTML 中插入 JavaScript 脚本需要使用<script>标签，浏览器会解释并执行位于<script>和</script>之间的 JavaScript 代码。

下面是几点常用的 JavaScript 功能，如写入 HTML 文档使用函数 document.write("写入内容")；对事件做出反应使用语句<button type="button" onclick=function()>单击</button>；改变 HTML 中某处的内容首先使用函数 document.getElementById("位置")查找要更改的位置，然后使用 innerHTML=""语句改写对应位置的内容。

JSP 全名为 Java Server Pages，中文名叫 Java 服务器页面。JSP 技术是在传统的网页 HTML 文件中插入 Java 程序段和 JSP 标记，从而形成 JSP 文件，扩展名为"*.jsp"。用 JSP 开发的 Web 应用是跨平台的，既能在 Linux 下运行也能在其他操作系统上运行。通常返回给客户端的就是一个 HTML 文本，因此客户端只要有浏览器就能浏览。用 JSP 技术，Web 页面开发人员可以使用 HTML 或者 XML 标识来设计和格式化最终页面，并使用 JSP 标识或者小脚本来生成页面上的动态内容。

上述就是客户端编写所用到的一些技术，整个客户端功能整体分为两部分，一部分是信息显示，一部分是管理配置，其中管理配置又分为针对单个设备的配置管理与选择相应策略对所有设备整体的管理。设备信息显示部分可以采用 JSP 技术，在 HTML 中嵌入 Java 程序脚本从数据库中取数据，动态显示在网页中。如果在网页中动态显示设备功率信息，代码示例如下所示，需要使用<% %>将 Java 代码括起来。

```
<%
Getinfo getinfo = newGetinfo("Power","db_user","password");
int power = getinfo.getmsg("select * from power order by time desc limit 1");
%>
```

在 HTML 文档中使用如下代码语句将从数据库中获取的数据显示到网页中：

```
<td>功率</td><td><% = power %></td>
```

对于单个设备参数的配置，使用 form 表单提交数据，如下代码所示是使用 form 表

单向服务器提交功率参数。

```
<form action = "Controller" method = "get" name = "form1">
            <td>功率</td>
            <td><input type = "text" name = "power"></td>
            <td><input type = "submit" class = "btn" value = "确定"></td>
</form>
```

其中，action 的属性值 Controller 表示所提交的 Servlet 类名，method 表示提交的方法，input type 表示输入信息元素的类型。当单击"确定"按钮，触发提交操作，数据以 get 的方式提交到名为 Controller 的 Servlet 类，Servlet 类接收以后将参数进行处理并传递到设备端。

对于网络整体管理策略实施，采用 Java 语言编写管理策略类与方法，策略说明在前端显示。用户可以根据相应信息选择合适的策略，将策略参数传入 Web 服务器相应的 Servlet 中，Servlet 根据参数实例化策略类执行对应方法。

前面只是就搭建平台所可能涉及的知识与平台架构设计做了简单的介绍，至于如何设计异构网络平台与采用什么技术实现还需要根据实际需求而定。

参考文献

[1] 屈在宏，章淼，徐明伟，等. 基于 XML 的互联网网络管理研究综述[J]. 小型微型计算机系统，2008，29(2)：245-250.
[2] 乐俊. 基于 NETCONF 的可扩展代理的研究[D]. 武汉：华中师范大学，2008.
[3] 李战国，唐亚哲，李增智. 分布式网络管理技术及实现[J]. 数据通信，2000(3)：23-25.
[4] 韦林. PHP 和 MySQL Web 开发[M]. 北京：机械工业出版社，2009.
[5] 孙卫琴. Tomcat 与 Java Web 开发技术详解[M]. 北京：电子工业出版社，2004.